

```

1 File: predictiondecisionengine/trunk/bin/aws_launch_benchmark.sh
2 #!/bin/bash
3 . /home/crivella/bin/functions.sh
4
5 #Check if needed packages are installed
6 if [[ `which aws` = "" || `which aws | grep -F "no aws"` != "" ]]; then
7     echo "This script require aws-cli to run."
8     echo "Follow the instructions on https://aws.amazon.com/cli/ on how to get it"
9     exit
10
11 fi
12 if [[ `which nmap` = "" || `which nmap | grep -F "no nmap"` != "" ]]; then
13     echo "This script require nmap to work. Install it by writing sudo apt-get install nmap"
14     echo "...Or the equivlaent command for your distribution"
15     exit
16 fi
17
18 re='^[0-9]+$'
19
20 #Edit this part with your data
21 #Where you have the run-* .sh files
22 FILES_PATH=$HOME/bin/Fermilab
23 #Where you stored your .pem keys
24 KEYS_PATH=$HOME/Fermilab/Keys
25 #where to keep the record of the machines you launched
26 TMP_PATH=$HOME/Fermilab/tmp
27
28 #Your username
29 USER=grassano
30 #ID of the AMI you will be using
31 AMI_ID=ami-03dec833
32 #Just the name with no .pem, or modify rest of the script
33 KEY=usw_oregon_grassano
34 #ID of the security group
35 SG_ID=sg-91d414f5
36 #Not the same for every instance
37 SAME_INSTANCE_LIMIT=25
38 #Bucket to get the benchmark files from

```

```

39 TEST_BUCKET=grassano-test
40
41 #Benchmark that the script can run
42 declare -a benchmarks=(tt_bar_gensim tt_bar_reco hepspec06 s3_stresstest_d s3_stresstest_u
... test_fermigrid test_fermigrid2)
43 #Volume to add with required size for the benchmark
44 declare -a req_vol_size=(80 80 50 0 3 3 3)
45 #Role
46 declare -a roles=("" "" "" "" AllowS3_Upload "")
47 declare -a parameters=("" "" "" "1 10 100" "1 10 100" "1 5 10 20" "1 5 10 20")
48
49 #Check for registered user
50 USER_DATA=`aws iam get-user`
51 USER_ARN=`echo "$USER_DATA" | grep Arn | cut -d ":" -f 4`
52 if [[ -z "$USER_ARN" ]]; then
53     echo "You don't appear to be registered to aws services. Exiting"
54 else
55     USER=`echo "$USER_DATA" | grep UserName | cut -d ":" -f 4`
56     USER_ID=`echo "$USER_ARN" | cut -d ":" -f 5`
57     #echo $USER:$USER_ID
58 fi
59
60 declare -a instances=(1:t2.micro)
61 RESUME_STATE="false"
62 ZONE=""
63 while getopts hra:s:k:i:p:z: opt; do
64     case $opt in
65         a) AMI_ID=$OPTARG
66             ;;
67         s) SG_ID=$OPTARG
68             ;;
69         k) KEY=$OPTARG
70             ;;
71         i) unset instances
72             C_COUNT=0
73             for CMD in $OPTARG; do
74                 if [[ ! `echo $CMD | cut -d ":" -f 1` =~ $re || `echo $CMD | grep ":" -o | wc -l` -gt 1
... ]]; then

```

```

75         echo "Invalid parameter for -i. Use the -h for more info."
76         exit
77     fi
78     instances[$C_COUNT]=$CMD
79     let C_COUNT++
80     done
81
82     ;;
83 ) PARAM=$OPTARG
84     ;;
85 z) if [[ `grep "profile $OPTARG" $HOME/.aws/config` != "" ]]; then
86     ZONE="--profile $OPTARG"
87 else
88     echo "You don't have a profile to work with this zone. Exiting..."
89     exit
90 fi
91     ;;
92 r) RESUME_STATE="true"
93 if [[ "$OPTARG" != "" ]]; then
94     echo "The -r does not take additional parameter. Use the -h for additional info"
95     exit
96 fi
97     ;;
98 h | \?) cat<<EOU
99
100 Usage: $0 [-a AMI_ID] [-s Security_group_ID] [-k KEY_NAME] [-i instances] [-r] [-p parameters]
101 [z zone]
102 -a AMI_ID           ID of the AMI to use
103 -s Security_group_ID   ID of the security group to use
104 -k KEY_NAME          Specify the name of the key to use without .pem
105 -i instances          List of instances to start es: "3:m3.medium 2:c4.2xlarge"
106 -p parameters        Parameters to pass to the run-* .sh script as "...."
107 -z zone              Region to launch the instances in
108 -r                  Enable resume mode
109 EOU
110     exit
111     ;;
112 esac

```

```

112 done
113
114 #Check if the selected AMI is available, if not give option to chose from list of owned
115 #available AMIs
116 while [[ `aws ec2 describe-images $ZONE --image-id $AMI_ID 2>&1 | grep "\"Name\":"` = "" ]]; do
117     echo "The specified ami does not exist. Do you wish to choose one from the ones you
118 own?(y/n)"
119     read_choice 5 "y" "n" "yes" "no"
120     if [[ `echo $CHOICE | grep y` = "" ]]; then
121         exit
122     else
123         IMAGES=`aws ec2 describe-images $ZONE --filters Name=state,Values=available --owners
124 $USER_ID`
125         IMAGES_NAME=`echo "$IMAGES" | grep "\"Name\":" | cut -d "\"" -f 4`
126         IMAGES_ID=`echo "$IMAGES" | grep "\"ImageId\":" | cut -d "\"" -f 4`
127         make_list "$IMAGES_NAME"
128         read_choice 5 `seq $LENGTH`
129         AMI_ID=`echo "$IMAGES_ID" | head -n $CHOICE | tail -n 1`
130         #echo $AMI_ID
131     fi
132     unset IMAGES
133     unset IMAGES_NAME
134     unset IMAGES_ID
135 done
136
137 #Check if the chosen key is available and recognized by aws
138 while [[ `find $KEYS_PATH -name "$KEY.pem"` = "" || `aws ec2 describe-key-pairs $ZONE | grep
139 "$KEY` = "" ]]; do
140     echo "You dont own the specified key or it is not recognized by amazon."
141     echo "Do you wish to choose it from the ones listed in aws that you own as well?(y/n)"
142     read_choice 5 "y" "n" "yes" "no"
143     if [[ `echo $CHOICE | grep y` = "" ]]; then
144         exit
145     else
146         KEYS=`find $KEYS_PATH -name "*.pem"`
147         KEYS_DATA=`aws ec2 describe-key-pairs $ZONE`
148         C_COUNT=0

```

```

145     while read -r line; do
146         line=${line::4}
147         FIELD=`echo $line | grep "/" -o | wc -l`
148         let FIELD++
149         KEY=`echo $line | cut -d "/" -f $FIELD`
150         if [[ `echo "$KEYS_DATA" | grep "$KEY"` != "" ]]; then
151             KEY_LIST[$C_COUNT]="$KEY"
152             let C_COUNT++
153         fi
154     done <<< "$KEYS"
155     if [[ $C_COUNT -lt 1 ]]; then
156         echo "No matching key found. Exiting..."
157         exit
158     fi
159     make_list "${KEY_LIST[@]}"
160     read_choice 5 `seq $LENGHT`
161     let CHOICE--
162     KEY="${KEY_LIST[$CHOICE]}"
163     #echo $KEY
164     fi
165     unset KEY_LIST
166     unset KEYS
167     unset KEYS_DATA
168 done
169
170 ##Check if the selected SG is available, if not give option to chose from list of availabe SGs
171 while [[ `aws ec2 describe-security-groups $ZONE --group-ids $SG_ID 2>&1 | grep
.. "\\"GroupId\":"` = "" ]]; do
172     echo "The specified security group does not exist.Do you wish to chose one frome those
.. available on aws?(y/n)"
173     read_choice 5 "y" "n" "yes" "no"
174     if [[ `echo $CHOICE | grep y` = "" ]]; then
175         exit
176     else
177         SGS=`aws ec2 describe-security-groups $ZONE`
178         SGS_NAME=`echo "$SGS" | grep "\"GroupName\":" | cut -d "\"" -f 4`
179         #echo "$SGS_NAME"
180         SGS_ID=`echo "$SGS" | grep "\"GroupId\":" | cut -d "\"" -f 4`
```

```

181     make_list "$SGS_NAME"
182     read_choice 5 `seq $LENGHT`
183     SG_ID=`echo "$SGS_ID" | head -n $CHOICE | tail -n 1`
184     #echo $SG_ID
185     fi
186     unset SGS
187     unset SGS_NAME
188     unset SGS_ID
189 done
190
191 mkdir -p $TMP_PATH/launch_logs
192
193 echo "Chose the kind of benchmarks to execute:"
194
195 make_list "${benchmarks[@]}"
196 read_choice 5 `seq $LENGHT`
197 let CHOICE--
198
199 benchmark=${benchmarks[$CHOICE]}
200 vol_size=${req_vol_size[$CHOICE]}
201
202 if [[ -z "$PARAM" ]]; then
203     PARAM=${parameters[$CHOICE]}
204 fi
205
206 if [[ $vol_size -eq 0 ]]; then
207     vol_size=3
208     for CMD in ${PARAM[@]}; do
209         let vol_size+=${CMD}
210     done
211 fi
212
213 #If a role is associated with the benchmark, enable it for the run-instances command
214 if [[ "${roles[$CHOICE]}" = "" ]]; then
215     ROLE="AllowS3_Download"
216 else
217     ROLE="${roles[$CHOICE]}"
218 fi
```

```

219
220 ROLE="--iam-instance-profile Name=\"$ROLE\""
221
222 #echo "$benchmark:$vol_size:$ROLE"
223 #{instances[@]}"
224
225 ######
226 #Check if the TEST_BUCKET in s3 contains a folder
227 #This script wont work if all the require files aren't in an s3 bucket in a folder with the
228 #Name set as the benchmark name
229 #Exception for s3_stresstest_d/u and test_fermigrid that go with the same foder s3_stresstest
230
231 if [[ `echo $benchmark | grep s3_stresstest` != "" || `echo "$benchmark" | grep
232 "test_fermigrid"` != "" ]]; then
233     a_benchmark="s3_stresstest"
234 else
235     a_benchmark=$benchmark
236 fi
237
238 if [ "`aws s3api list-objects --bucket $TEST_BUCKET | grep $a_benchmark`" = "" ]; then
239     echo "Can't find the right folder in s3. Quitting..."
240     exit
241 fi
242
243 #echo -e "$instances\n$PARAM\n$benchmark:$vol_size"
244 #exit
245
246 ######
247 #Create folder to store the VMS data + Resume work option
248 COUNT2=0
249 RESUME=0
250 RES_STEP=0
251 mkdir -p $TMP_PATH/$benchmark
252 if $RESUME_STATE; then
253     if [[ ! ${#instances[@]} -eq 1 ]]; then
254         echo "The resume functionality works only with one kind of instances and benchmark."

```

```

253     exit
254 fi
255 if [[ ! -f $TMP_PATH/$benchmark/VMS ]]; then
256     echo "Tmp file not found. Exiting..."
257     exit
258 fi
259 while read -r line; do
260     if [[ "$line" != "" ]]; then
261         I_CHECK=`echo ${instances[0]} | cut -d ":" -f 2`
262         I_TYPES[$COUNT2]=`echo "$line" | cut -d ":" -f 2`
263         if [[ "${I_TYPES[$COUNT2]}" != "$I_CHECK" ]]; then
264             echo "The specified instance type does not match the ones in the tmp file.
...Can't use the resume functionality"
265             exit
266         fi
267         VM_IDS[$COUNT2]=`echo "$line" | cut -d ":" -f 3`
268         let COUNT2++
269         let RESUME++
270     fi
271 done < $TMP_PATH/$benchmark/VMS
272
273 for ((COUNT=0,COUNT<COUNT2,COUNT++)); do
274     INSTANCE_DATA=`aws ec2 describe-instances $ZONE --instance-ids ${VM_IDS[$COUNT]}``$VM_AVAL[$COUNT]=`echo "$INSTANCE_DATA" | grep -F "AvailabilityZone" | cut -d "\n" -f
275 4`$VM_PUB_DNS[$COUNT]=`echo "$INSTANCE_DATA" | grep PublicDnsName | cut -d "\n" -f 4 |`$uniq`$ssh -o "StrictHostKeyChecking no" -i $KEYS_PATH/$KEY.pem root@${VM_PUB_DNS[$COUNT]}`$rm -f -r /scratch/*">>/dev/null 2>&1 &
276     if [[ `echo "$INSTANCE_DATA" | grep running` = "" ]]; then
277         echo "One or more of the instances described in the tmp file do not exist anymore
...or are not running."
278         echo "Can't continue with the resume functionality"
279         exit
280     fi
281 done
282 unset INSTANCE_DATA
283

```

```

286     step="Volume creation
287         Start Benchmark"
288     echo "Select the step from which you want to resume the work for the machine stored in
289     ... tmp:"
290     make_list "$step"
291     read_choice 3 `seq $LENGTH`
292     RES_STEP=$CHOICE
293     unset step
294 else
295     rm -f $TMP_PATH/$benchmark/VMS
296 fi
297 #####
298 #####
299 #Starting the required VM
300 STEP=0
301 for ((COUNT=0;COUNT<${#instances[@]};COUNT++)); do
302     I_NUM=`echo "${instances[$COUNT]}" | cut -d ":" -f 1`
303     I_TYPE=`echo "${instances[$COUNT]}" | cut -d ":" -f 2`
304
305     if [[ $RESUME -gt 0 ]]; then
306         let I_NUM-= $RESUME
307         I_COUNT=$((expr $RESUME + 1))
308     else
309         I_COUNT=1
310     fi
311     if [[ ${I_NUM} -gt ${SAME_INSTANCE_LIMIT} ]]; then
312         echo "The number (${I_NUM}) specified for instance ${I_TYPE} exceeds the limit of
313         ${SAME_INSTANCE_LIMIT}."
314         echo "Bringing it down to the maximum allowed..."
315         I_NUM=${SAME_INSTANCE_LIMIT}
316     fi
317     if [[ ${I_NUM} -gt 0 ]]; then
318         echo "Launching ${I_NUM} ${I_TYPE} instances"
319         INSTANCE_DATA=`aws ec2 run-instances $ZONE --count ${I_NUM} --image-id ${AMI_ID}
320         --instance-type ${I_TYPE} --security-group-ids ${SG_ID} --key-name ${KEY} ${ROLE}`
```

```

320     fi
321
322     L_COUNT=1
323     while [[ ${L_COUNT} -le ${I_NUM} ]]; do
324         I_TYPES[$COUNT2]=${I_TYPE}
325         INSTANCE_ID=`echo "$INSTANCE_DATA" | grep -F "InstanceId" | head -n ${L_COUNT} | tail -n
326         -1 | cut -d "\"" -f 4`
327         VM_IDS[$COUNT2]=${INSTANCE_ID}
328         AVAL=`echo "$INSTANCE_DATA" | grep -F "AvailabilityZone" | head -n ${L_COUNT} | tail -n
329         -1 | cut -d "\"" -f 4`
330         VM_AVAL[$COUNT2]=${AVAL}
331         aws ec2 create-tags ${ZONE} --resources ${INSTANCE_ID} --tags
332         Key=Name,Value=${USER}-${I_TYPE}-$benchmark` printf %03d ${I_COUNT}` Key=User,Value=${USER}>/dev/null
333         let I_COUNT++
334         let L_COUNT++
335         let COUNT2++
336     done
337     done
338 LAST_INSTANCE=$((expr ${COUNT2} - 1))
339 #####
340 #Get the public DNS after they are available
341 echo "Waiting for the DNS..."
342 while [[ `aws ec2 describe-instances $ZONE --instance-ids ${VM_IDS[$LAST_INSTANCE]} | grep -F
343         "PublicDnsName" | cut -d "\"" -f 4 | uniq` = "" ]]; do
344     sleep 10
345 done
346 sleep 5
347 #echo "$RESUME:${#VM_IDS[@]}"
348 for ((COUNT=$RESUME;COUNT<${#VM_IDS[@]};COUNT++)); do
349     INSTANCE_DATA=`aws ec2 describe-instances $ZONE --instance-ids ${VM_IDS[$COUNT]}`
350     PUB_DNS=`echo "$INSTANCE_DATA" | grep -F "PublicDnsName" | cut -d "\"" -f 4 | uniq`  

351     #Print VMS data in tmp file
352     echo "$COUNT:${I_TYPES[$COUNT]}:${VM_IDS[$COUNT]}:$PUB_DNS:$KEY" >>
353     $TMP_PATH/$benchmark/VMS
```

```

352     VM_PUB_DNS[$COUNT]="$PUB_DNS"
353 done
354
355 ##### Create extra volumes #####
356 #Create extra volumes
357 let STEP++
358 if [[ $STEP -ge $RES_STEP ]]; then
359     RES_VAL=0
360 else
361     RES_VAL=$RESUME
362     I_COUNT=$(expr $RESUME + 1)
363 fi
364
365 for ((COUNT=$RES_VAL;COUNT<${#VM_IDS[@]};COUNT++)); do
366     if [[ "${I_TYPES[$COUNT]}" = "${I_TYPES[$(expr $COUNT - 1)]}" && $COUNT -gt $RES_VAL ]]; then
367         let I_COUNT++
368     else
369         I_COUNT=$(expr $RES_VAL + 1)
370     fi
371     echo "Creating extradisk for instance ${I_TYPES[$COUNT]}-`printf %03d $I_COUNT`"
372     VOLUME_DATA=`aws ec2 create-volume $ZONE --size $vol_size --availability-zone
373     ${VM_AVAL[$COUNT]} --volume-type gp2`
374     VOLUME_ID[$COUNT]=`echo "$VOLUME_DATA" | grep -F "VolumeId" | cut -d "\"" -f 4`"
375 done
376 #####
377 #Wait until all VM are done initialazing
378 echo "Waiting for the instances to be initialized..."
379 for ((COUNT=$RES_VAL;COUNT<${#VM_IDS[@]};COUNT++)); do
380     while [[ `nmap -p 22 ${VM_PUB_DNS[$COUNT]} | grep open` = "" ]]; do
381         sleep 15
382     done
383 done
384
385

```

```

386 #####
387 #Disable the self-stop service and register the connection to the DNS (won't ask yes/no
388 # afterwards)
389 for ((COUNT=0;COUNT<${#VM_IDS[@]};COUNT++)); do
390     ssh -o "StrictHostKeyChecking no" -i $KEYS_PATH/$KEY.pem root@${VM_PUB_DNS[$COUNT]}
391     "service glideinwms-pilot stop">/dev/null
392     #echo "Stopped service in instance ${instances[$COUNT]}."
393 done
394 sleep 3
395 #####
396 #Waiting for volumes to create
397 echo "Waiting for volumes to create..."
398 for ((COUNT=$RES_VAL;COUNT<${#VM_IDS[@]};COUNT++)); do
399     while [[ `aws ec2 describe-volumes $ZONE --volume-ids ${VOLUME_ID[$COUNT]} | grep
400     available` = "" ]]; do
401         #echo "Waiting for disk to create..."
402         sleep 5
403     done
404     if [[ "${I_TYPES[$COUNT]}" = "${I_TYPES[$(expr $COUNT - 1)]}" && $COUNT -gt $RES_VAL ]]; then
405         let I_COUNT++
406     else
407         I_COUNT=$(expr $RES_VAL + 1)
408     fi
409     aws ec2 create-tags $ZONE --resources ${VOLUME_ID[$COUNT]} --tags
410     Key=$NAME,Value=$USER-$I_TYPES[$COUNT]-$benchmark`printf %03d $I_COUNT`-
411     Key=user,Value=$USER>/dev/null
412 done
413 #####
414 #Attaching Volumes
415 echo "Attaching Volumes..."
416 for ((COUNT=$RES_VAL;COUNT<${#VM_IDS[@]};COUNT++)); do
417     aws ec2 attach-volume $ZONE --volume-id ${VOLUME_ID[$COUNT]} --instance-id

```

```

414. ${VM_IDS[$COUNT]} --device /dev/sdf>/dev/null
415 done
416
417 ######
418 #Waiting for volumes to attach
419 echo "Waiting for volumes to attach and making filesystem..."
420 for ((COUNT=$RES_VAL;COUNT<${#VM_IDS[@]};COUNT++)); do
421     while [[ `aws ec2 describe-volumes $ZONE --volume-ids ${VOLUME_ID[$COUNT]} | grep
422     attaching` != "" ]]; do
423         sleep 3
424     done
425     if [[ "${I_TYPES[$COUNT]}" = "${I_TYPES[$(expr $COUNT - 1)]}" ]] && $COUNT -gt $RES_VAL ]]; then
426         let I_COUNT++
427     else
428         I_COUNT=$(expr $RES_VAL + 1)
429     fi
430     echo "Making file system ext4 for ${I_TYPES[$COUNT]}-$I_COUNT..."
431     ssh -i $KEYS_PATH/$KEY.pem root@$VM_PUB_DNS[$COUNT] "mkfs.ext4 /dev/xvdf
432     2>/dev/null">/dev/null &
433 done
434 wait
435 #####
436 #Start the benchmark
437 let STEP++
438 if [[ $STEP -ge $RES_STEP ]]; then
439     RES_VAL=0
440 else
441     RES_VAL=$RESUME
442 fi
443 echo "Starting the benchmark on every instance..."
444 TIME=`date +%R`
445 for ((COUNT=$RES_VAL;COUNT<${#VM_IDS[@]};COUNT++)); do
446     scp -i $KEYS_PATH/$KEY.pem $FILES_PATH/run-$benchmark.sh
447     root@$VM_PUB_DNS[$COUNT]:/root/>/dev/null 2>&1

```

```

446     ssh -i $KEYS_PATH/$KEY.pem root@$VM_PUB_DNS[$COUNT] "chmod 755 /root/run-$benchmark.sh"
447     >/dev/null 2>&1
448     echo
449     #####"
450     #####" >>$TMP_PATH/launch_logs/run_error_`printf %03d
451     $COUNT`.log
452     echo ``date` ${VM_IDS[$COUNT]}:${I_TYPES[$COUNT]}:$benchmark"
453     >>$TMP_PATH/launch_logs/run_error_`printf %03d $COUNT`.log
454     ssh -i $KEYS_PATH/$KEY.pem root@$VM_PUB_DNS[$COUNT] "/root/run-$benchmark.sh $TIME
455     $COUNT $PARAM">/dev/null 2>>$TMP_PATH/launch_logs/run_error_`printf %03d $COUNT .log 2>&1 &
456     done
457
458 echo "I'm done. Wait for the benchmark to be over and than launch the crop_results.sh script."
459
460 exit
461
462 -----
463 -----
464 -----
465
466 File: predictiondecisionengine/trunk/bin/aws_login.sh
467 #!/bin/bash
468 . /home/crivella/bin/functions.sh
469
470 while getopts hz: opt; do
471     case $opt in
472         z) if [[ `grep "profile $OPTARG" $HOME/.aws/config` != "" ]]; then
473             ZONE="--profile $OPTARG"
474         else
475             echo "You don't have a profile to work with this zone. Exiting..."
476             exit
477         fi
478         ;;
479         h | \?) cat<<EOU

```

```

478 Usage: $0 [z zone]
479   -z zone           Region to launch the instances in
480 EOU
481     exit
482   ;;
483   esac
484 done
485
486 FILES_PATH=/home/crivella/bin/Fermilab
487 KEYS_PATH=/home/crivella/Fermilab
488 USER=
489
490 #Check for registered user
491 USER_DATA=`aws iam get-user`
492 USER_ARN=`echo "$USER_DATA" | grep Arn | cut -d "\"" -f 4`
493 if [[ -z "$USER_ARN" ]]; then
494   echo "You don't appear to be registered to aws services. Exiting"
495 else
496   USER=`echo "$USER_DATA" | grep UserName | cut -d "\"" -f 4`
497   USER_ID=`echo "$USER_ARN" | cut -d ":" -f 5`
498   #echo $USER:$USER_ID
499 fi
500
501 VMS_DATA=`aws ec2 describe-instances $ZONE --filters Name=tag-value,Values=$USER
... Name=instance-state-name,Values=running`
502 if [[ `echo "$VMS_DATA" | grep $USER` = "" ]]; then
503   echo "No running instances found for user $USER. Quitting script..."
504   exit
505 fi
506
507 VMS=`echo "$VMS_DATA" | grep -F "$USER-" | cut -d "\"" -f 4`
508 A_VMS=(${list_to_array "$VMS"})
509 unset VMS
510 VMS_KEYS=`echo "$VMS_DATA" | grep -F "_$USER" | cut -d "\"" -f 4`
511 A_VMS_KEYS=(${list_to_array "$VMS_KEYS"})
512 unset VMS_KEYS
513 VMS_PUB_DNS=`echo "$VMS_DATA" | grep -F "PublicDnsName" | cut -d "\"" -f 4 | uniq`
514 A_VMS_PUB_DNS=(${list_to_array "$VMS_PUB_DNS"})

```

```

515 unset VMS_PUB_DNS
516
517 #Sort the VMs by name while also doing the same operation on the VMS_KEYS and VMS_PUB_DNS
... arrays
518 for ((i=0;i<#${A_VMS[@]}-1;i++)); do
519   for ((j=i+1;j<#${A_VMS[@]};j++)); do
520     if [[ "${A_VMS[$i]}" > "${A_VMS[$j]}" ]]; then
521       APP=${A_VMS[$i]}
522       A_VMS[$i]=${A_VMS[$j]}
523       A_VMS[$j]=$APP
524       APP=${A_VMS_KEYS[$i]}
525       A_VMS_KEYS[$i]=${A_VMS_KEYS[$j]}
526       A_VMS_KEYS[$j]=$APP
527       APP=${A_VMS_PUB_DNS[$i]}
528       A_VMS_PUB_DNS[$i]=${A_VMS_PUB_DNS[$j]}
529       A_VMS_PUB_DNS[$j]=$APP
530     fi
531   done
532 done
533
534 make_list "${A_VMS[@]}"
535 read_choice 5 `seq $LENGTH`
536 let CHOICE--
537
538 VM=${A_VMS[$CHOICE]}
539 VM_DNS=${A_VMS_PUB_DNS[$CHOICE]}
540 VM_KEY=${A_VMS_KEYS[$CHOICE]}
541
542 KEY=`find $KEYS_PATH -name "$VM_KEY.pem"`
543 if [ "$KEY" = "" ]; then
544   echo "You do not have the necessary key. Terminating..."
545   exit
546 fi
547
548 :
549 echo "Do you want to copy the files?[y/n]: "
550 read ANSWER
551 if [[ $ANSWER = "y" ]]; then

```

```

552     scp -i $KEY $FILES_PATH/runttbar.sh root@$VM_DNS:/root/
553     scp -i $KEY $FILES_PATH/res_show.sh root@$VM_DNS:/root/
554     scp -i $KEY $FILES_PATH/runhepspec.sh root@$VM_DNS:/root/
555     #scp -i $KEY $FILES_PATH/runparallel.sh root@$VM_DNS:/root/
556 fi'
557
558 ssh -i $KEY root@$VM_DNS
559
560 -----
561 -----
562 -----
563
564
565 File: predictiondecisionengine/trunk/bin/crop_results.sh
566 #!/bin/bash
567 . /home/crivella/bin/functions.sh
568
569
570 #Check if needed packages are installed
571 if [[ `which aws` = "" || `which aws | grep -F "no aws"` != "" ]]; then
572     echo "This script require aws-cli to run."
573     echo "Follow the instructions on https://aws.amazon.com/cli/ on how to get it"
574     exit
575
576 fi
577 if [[ `which nmap` = "" || `which nmap | grep -F "no nmap"` != "" ]]; then
578     echo "This script require nmap to work. Install it by writing sudo apt-get install nmap"
579     echo "...Or the equivalent command for your distribution"
580     exit
581 fi
582
583 #Edit this part with your data
584 #Path where you wish to store the results of the benchmarks
585 B_PATH=/home/crivella/Fermilab/Benchmarks
586 #Your username
587 USER=grassano
588 #Where you have the .sh files
589 FILES_PATH=/home/crivella/bin/Fermilab

```

```

590 #Where you stored your .pem keys
591 KEYS_PATH=/home/crivella/Fermilab/Keys
592 #They name of the key you used without the .pem
593 KEY=usw_oregon_grassano
594 #where to keep the record of the machines you launched
595 TMP_PATH=/home/crivella/Fermilab/tmp
596
597 while getopts hk:z: opt; do
598     case $opt in
599         z)  if [[ `grep "profile $OPTARG" $HOME/.aws/config` != "" ]]; then
600             ZONE="--profile $OPTARG"
601         else
602             echo "You don't have a profile to work with this zone. Exiting..."
603             exit
604         fi
605         ;;
606         k)  KEY=$OPTARG
607         ;;
608         h | \?) cat<<EOU
609
610 Usage: $0 [-k KEY_NAME] [z zone]
611       -k KEY_NAME           Specify the name of the key to use without .pem
612       -z zone               Region to launch the instances in
613 EOU
614         exit
615         ;;
616     esac
617 done
618
619 declare -a benchmarks=(tt_bar_gensim tt_bar_reco hepspec06 s3_stresstest_d s3_stresstest_u
620 ... test_fermigrid test_fermigrid2)
621
622 echo "Chose the kind of benchmarks to crop the results for:"
623 make_list "${benchmarks[@]}"
624 read_choice 5 `seq ${#benchmarks[@]}`
625 let CHOICE--
626

```

```

627 ||benchmark=${benchmarks[$CHOICE]}
628 mkdir -p $B_PATH/$benchmark
629
630 if [[ -d $TMP_PATH/$benchmark ]]; then
631   echo "I've found a tmp folder for the benchmark. Do you wish to use VM data from here?(y)"
632   echo "... Or do yo wish to query amazon for active VMS? (n)"
633   read_choice 5 "y" "n" "yes" "no"
634   FLAG=1
635 fi
636
637 if [[ "$CHOICE" != "n" && FLAG -eq 1 ]]; then
638   while read -r line; do
639     POS=`echo "$line" | cut -d ":" -f 1`
640     VMS[$POS]=$(`echo "$line" | cut -d ":" -f 2`)
641     VM_IDS[$POS]=$(`echo "$line" | cut -d ":" -f 3`)
642     VM_PUB_DNS[$POS]=$(`echo "$line" | cut -d ":" -f 4`)
643     VM_KEYS[$POS]=$(`echo "$line" | cut -d ":" -f 5`)
644   done < $TMP_PATH/$benchmark/VMS
645 else
646   #Adjust this section with your own format used to name instances and keys#####
647   VMS_DATA=`aws ec2 describe-instances --filters Name=tag-value,Values=$USER`
648   VMS_NAME=`echo "$VMS_DATA" | grep -F "$USER-" | cut -d "\\" -f 4`
649   PUB_DNS=`echo "$VMS_DATA" | grep -F "PublicDnsName" | cut -d "\\" -f 4 | uniq`
650   VMS_KEYS=`echo "$VMS_DATA" | grep -F "$USER" | cut -d "\\" -f 4`
651 #####
652
653   VM_PUB_DNS=(${list_to_array "$PUB_DNS"})
654   VMS=(${list_to_array "$VMS_NAME"})
655   VM_KEYS=(${list_to_array "$VMS_KEYS"})
656   unset VMS_DATA
657   unset PUB_DNS
658   unset VMS_KEYS
659 fi
660
661 if [[ "${VM_PUB_DNS[0]}" = "" ]]; then
662   echo "No results found..."
663   exit
664 fi

```

```

665 :
666 :
667 for ((COUNT=0;COUNT<${#VM_PUB_DNS[@]};COUNT++)); do
668   echo "$COUNT"
669   echo "${VMS[$COUNT]}"
670   echo "${VM_IDS[$COUNT]}"
671   echo "${VM_PUB_DNS[$COUNT]}"
672 done
673 exit'
674
675 if [[ "$benchmark" = "tt_bar_reco" || "$benchmark" = "tt_bar_gensim" ]]; then
676 cat <<CROP > crop.sh
677 #!/bin/bash
678
679 cd /tmp/cms
680 FOLDERS=\`ls | grep results | grep -v .txt | grep -v .sh\``
681
682 for CMD in \$FOLDERS; do
683   if [[ -e check_results.sh ]]; then
684     sh check_results.sh > $CMD.txt
685   else
686     cd \$CMD
687     sh ../check_results_gensim.sh > ../\$CMD.txt
688     cd ..
689   fi
690 done
691 exit
692 CROP
693 fi
694
695 for ((COUNT=0;COUNT<${#VM_PUB_DNS[@]};COUNT++)); do
696   if [[ (`echo "$benchmark" | grep "s3_stresstest"`) = "" && `echo "$benchmark" | grep "test_fermigrid"` = "" || $COUNT -eq 0 ]];
697   || ((`echo "$benchmark" | grep "s3_stresstest"`) != "" && `echo "$benchmark" | grep "test_fermigrid"`) != "" && $COUNT -gt 0 && "${VMS[$COUNT]}" != "${VMS[\$(expr \$COUNT - 1)]}"]; then
698     F_PATH="$B_PATH/$benchmark/${VMS[$COUNT]}"
699     A_PATH=$F_PATH
700     C_COUNT=0

```

```

700         while (! mkdir $A_PATH 2>/dev/null); do
701             let C_COUNT++
702             A_PATH=$F_PATH.run`printf %03d $C_COUNT`
703         done
704         F_PATH=$A_PATH
705     fi
706     #echo $F_PATH
707     if [[ ("$benchmark" = "tt_bar_reco" || "$benchmark" = "tt_bar_gensim") && $CHECK -eq 0 ]];
708     then
709         scp -i $KEYS_PATH/${VM_KEYS[$COUNT]}.pem crop.sh
710         root@$VM_PUB_DNS[$COUNT]:/root>/dev/null
711         ssh -i $KEYS_PATH/${VM_KEYS[$COUNT]}.pem root@$VM_PUB_DNS[$COUNT] "sh
712         /root/crop.sh">/dev/null
713         #ssh -i $KEYS_PATH/${VM_KEYS[$COUNT]}.pem root@$VM_PUB_DNS[$COUNT] "rm
714         /root/crop.sh">/dev/null
715         scp -i $KEYS_PATH/${VM_KEYS[$COUNT]}.pem
716         root@$VM_PUB_DNS[$COUNT]:/tmp/cms/results*txt $F_PATH>/dev/null
717         elif [[ "$benchmark" = "hepspec06" ]]; then
718             #rm -f crop.sh
719             scp -i $KEYS_PATH/${VM_KEYS[$COUNT]}.pem -r
720             root@$VM_PUB_DNS[$COUNT]:/scratch/install/results/* $F_PATH
721             #scp -r root@$VM_PUB_DNS[$COUNT]:/scratch/new_hepspec2006/install/results/* $F_PATH
722             #exit
723             RESULTS=`find $F_PATH -name "lock.CPU2006" | sort`
724             if [[ "$RESULTS" = "" ]]; then
725                 echo "The results folder is empty..."
726                 exit
727             fi
728             while read -r CMD; do
729                 RES_PATH=`dirname $CMD`
730                 N_CORE=`find $RES_PATH -name "CPU2006*.log" | wc -l`
731                 for COUNT in `seq $N_CORE`; do
732                     if [ $COUNT -eq 1 ]; then
733                         echo "CORE$COUNT" > $RES_PATH/crop_core.txt
734                     else
735                         echo -e "\nCORE$COUNT" >> $RES_PATH/crop_core.txt
736                     fi
737                     cat $RES_PATH/CFP2006.`printf %03d $COUNT`.ref.txt | grep -F "*" | sort |

```

```

731...    uniq >> $RES_PATH/crop_core.txt
732            cat $RES_PATH/CINT2006.`printf %03d $COUNT`.ref.txt | grep -F "*" | sort |
733        ... uniq >> $RES_PATH/crop_core.txt
734        ... done
735        ... done <<< "$RESULTS"
736        ... elif [[ `echo "$benchmark" | grep "s3_stresstest"` != "" || `echo "$benchmark" | grep
737        ... "test_fermigrid"` != "" ]]; then
738            rm -f crop.sh
739            if [[ "${VMS[$COUNT]}" = "${VMS[$(expr $COUNT - 1)]}" && $COUNT -gt 0 ]]; then
740                let I_COUNT++
741            else
742                I_COUNT=1
743            fi
744            if [[ `echo "$benchmark" | grep "test_fermigrid"` != "" ]]; then
745                scp -i $KEYS_PATH/${VM_KEYS[$COUNT]}.pem
746                root@$VM_PUB_DNS[$COUNT]:/scratch/fermigrid_upload.log $F_PATH/${VMS[$COUNT]}-$I_COUNT.log
747                scp -i $KEYS_PATH/${VM_KEYS[$COUNT]}.pem
748                root@$VM_PUB_DNS[$COUNT]:/scratch/command.log $F_PATH/command-$I_COUNT.log
749                else
750                    scp -i $KEYS_PATH/${VM_KEYS[$COUNT]}.pem
751                    root@$VM_PUB_DNS[$COUNT]:/scratch/s3_stress.log $F_PATH/${VMS[$COUNT]}-$I_COUNT.log
752                fi
753                RESULTS="$F_PATH/${VMS[$COUNT]}-$I_COUNT.log"
754                while read -r CMD; do
755                    RES_PATH=`dirname $CMD`
756                    string=`cat $CMD | grep -F ":"`  

757                    #echo "$string"
758                    C_COUNT=0
759                    string2=`cat $CMD | grep -F "Simultaneous" | cut -d " " -f 4`  

760                    #echo "$string2"
761                    N_COUNT=0
762                    while read -r APP; do
763                        NUM[$N_COUNT]=$APP
764                        let N_COUNT++
765                    done <<< "$string2"
766                    string2=`cat $CMD | grep -F "out of" | cut -d " " -f 1`  

767                    #echo "$string2"
768                    N_COUNT=0

```

```

764         while read -r APP; do
765             SUCC_NUM[$N_COUNT]=$APP
766             let N_COUNT++
767             done <<< "$string2"
768             N_COUNT=0
769             echo "$CMD" >> $RES_PATH/diff.log
770             while read -r line; do
771                 A_HOURS[$C_COUNT]=`echo "$line" | cut -d ":" -f 1`
772                 A_HOURS[$C_COUNT]=${A_HOURS[$C_COUNT]: -2}
773                 A_MIN[$C_COUNT]=`echo "$line" | cut -d ":" -f 2`
774                 A_MIN[$C_COUNT]=${A_MIN[$C_COUNT]: -2}
775                 A_SEC[$C_COUNT]=`echo "$line" | cut -d ":" -f 3`
776                 #echo ${A_SEC[$C_COUNT]}
777                 A_SEC[$C_COUNT]=${A_SEC[$C_COUNT]:2}
778                 #echo ${A_SEC[$C_COUNT]}
779                 let C_COUNT++
780                 if [[ $C_COUNT -ge 2 ]]; then
781                     H_DIFF=$(expr ${A_HOURS[1]} - ${A_HOURS[0]})  

782                     M_DIFF=$(expr ${A_MIN[1]} - ${A_MIN[0]})  

783                     S_DIFF=$(expr ${A_SEC[1]} - ${A_SEC[0]})  

784                     T_DIFF=`echo "$H_DIFF*3600+$M_DIFF*60+$S_DIFF" | bc`  

785                     if [[ "${NUM[$N_COUNT]}" = "" ]]; then  

786                         NUM[$N_COUNT]=$(NUM[$((N_COUNT-1))])  

787                     fi
788                     RATIO0=`echo "${NUM[$N_COUNT]}/$T_DIFF" | bc -l`  

789                     RATIO0=${RATIO0:4}  

790                     echo -ne "Diff for ${NUM[$N_COUNT]} is\t\t$T_DIFF in seconds\t\t$RATIO0  

791                     ... GB/s\t ${SUCC_NUM[$N_COUNT]} out of ${NUM[$N_COUNT]} up/downloads succeeded" >>  

792                     $RES_PATH/diff.log
793                     if [[ ${SUCC_NUM[$N_COUNT]} -lt ${NUM[$N_COUNT]} ]]; then  

794                         echo "<<<<<<<<<<<<<<<" >> $RES_PATH/diff.log
795                     else
796                         echo "" >> $RES_PATH/diff.log
797                     fi
798                     let N_COUNT++
799                     C_COUNT=0
800                 fi
801             done <<< "$string"

```

```

800             echo -e "\n" >> $RES_PATH/diff.log
801             done <<< "$RESULTS"
802         fi
803     done
804
805 #rm -f crop.sh
806 exit
807 ${NUM[$N_COUNT]}
808 -----
809 -----
810 -----
811 -----
812
813
814 File: predictiondecisionengine/trunk/bin/fc_login.sh
815#!/bin/bash
816
817 M_LOGIN_PATH=/home/crivella/bin/Fermilab
818 USER=grassano
819
820
821 echo "Are you already logged into kinit[y/n]: "
822 read ANSWER
823 if [[ $ANSWER = "n" ]]; then
824     kinit $USER@FNAL.GOV
825 fi
826
827
828 echo "Do you want to copy m_login?[y/n]: "
829 read ANSWER
830 if [[ $ANSWER = "y" ]]; then
831     scp $M_LOGIN_PATH/m_login.sh $USER@fcluigpvm01.fnal.gov:/cloud/login/$USER/
832 fi
833
834
835 ssh -l $USER fcluigpvm01.fnal.gov
836 -----
837 -----

```

```

838 |-----
839 |-----
840 |
841 |
842 File: predictiondecisionengine/trunk/bin/Fermilab/From_EBS/aws_launch_benchmark.sh
843 #!/bin/bash
844
845 #Description: Read the choice to a query and confront it with a list of correct answer
846 #           .. limiting the numbers of checks
847 #Parameters: 1st parameter: number of attempts possible other parameter:List of correct
848 #           .. answers
849 function read_choice()
850 {
851     local NUM=$1
852     shift
853     CHOICES=("${!#}")
854     while "true"; do
855         read CHOICE
856         CHOICE=$(echo $CHOICE | tr YESNO yesno`)
857         for CMD in ${CHOICES[@]}; do
858             if [[ "$CHOICE" == "$CMD" ]]; then
859                 return 0
860             fi
861         done
862         echo "Choice is out of range. Enter again :"
863         let C_COUNT++
864         if [[ $C_COUNT -ge $NUM ]]; then
865             echo "$NUM invalid choices. Exiting..."
866             exit
867         fi
868     done
869 }
870
871 #Description: Make a numbered list out of an array or list of elements and return the number
872 #           .. of elements in LENGTHT
873 #Parameters: array or list of elements
874 function make_list()
875 {

```

```

876     local C_COUNT=0
877     if [[ `echo "$1" | wc -l` -gt 1 ]]; then
878         echo Lista
879         while read -r line; do
880             let C_COUNT++
881             echo -e "$C_COUNT-\t$line"
882         done <<< "$1"
883     else
884         local array=("$@")
885         for CMD in ${array[@]}; do
886             let C_COUNT++
887             echo -e "$C_COUNT-\t$CMD"
888         done
889     fi
890     LENGTH=$C_COUNT
891 }
892
893 #Check if needed packages are installed
894 if [[ `which aws` = "" || `which aws | grep -F "no aws"` != "" ]]; then
895     echo "This script require aws-cli to run."
896     echo "Follow the instructions on https://aws.amazon.com/cli/ on how to get it"
897     exit
898
899 fi
900 if [[ `which nmap` = "" || `which nmap | grep -F "no nmap"` != "" ]]; then
901     echo "This script require nmap to work. Install it by writing sudo apt-get install nmap"
902     echo "...Or the equivalent command for your distribution"
903     exit
904
905 re='^[0-9]+$'
906
907 #Edit this part with your data
908 #Where you have the run-* .sh files
909 FILES_PATH=$HOME/bin/Fermilab
910 #Where you stored your .pem keys
911 KEYS_PATH=$HOME/Fermilab/Keys
912 #Where to keep the record of the machines you launched

```

```

911 TMP_PATH=$HOME/Fermilab/tmp
912
913 #Your username
914 USER=grassano
915 #ID of the AMI you will be using
916 AMI_ID=ami-03dec833
917 #Just the name with no .pem, or modify rest of the script
918 KEY=usw_oregon_grassano
919 #ID of the security group
920 SG_ID=sg-91d414f5
921 #Not the same for every instance
922 SAME_INSTANCE_LIMIT=25
923
924 #Benchmark that the script can run
925 declare -a benchmarks=(tt_bar_gensim tt_bar_reco hepspec06 s3_stresstest_d s3_stresstest_u
926 ... test_fermigridd)
927 #Volume to add with required size for the benchmark
928 declare -a req_vol_size=(80 80 50 120 3 3)
929 #Role related profile
930 declare -a profiles=("" "" "" AllowS3_Download AllowS3_Upload "AllowS3_Download")
931 declare -a parameters=("" "" "" "1 10 100" "1 10 100" "1 5 10 20")
932
933 #Check for registered user
934 USER_DATA=`aws iam get-user`
935 USER_ARN=`echo "$USER_DATA" | grep Arn | cut -d ":" -f 4`
936 if [[ -z "$USER_ARN" ]]; then
937     echo "You don't appear to be registered to aws services. Exiting"
938 else
939     USER=`echo "$USER_DATA" | grep UserName | cut -d ":" -f 4`
940     USER_ID=`echo "$USER_ARN" | cut -d ":" -f 5`
941     #echo $USER:$USER_ID
942 fi
943
944 declare -a instances=(1:r3.xlarge 1:r3.2xlarge 1:r3.4xlarge)
945 RESUME_STATE="false"
946 while getopts hra:s:k:i:p: opt; do
947     case $opt in
948         a) AMI_ID=$OPTARG

```

```

949         ;;
950         s) SG_ID=$OPTARG
951         ;;
952         K) KEY=$OPTARG
953         ;;
954         i) unset instances
955             C_COUNT=0
956             for CMD in $OPTARG; do
957                 if [[ ! `echo $CMD | cut -d ":" -f 1` =~ $re || `echo $CMD | grep ":" -o | wc -l` -gt 1
958             ]]; then
959                 echo "Invalid parameter for -i. Use the -h for more info."
960                 exit
961             fi
962             instances[$C_COUNT]=$CMD
963             let C_COUNT++
964             done
965         ;;
966         p) PARAM=$OPTARG
967         ;;
968         r) RESUME_STATE="true"
969         if [[ "$OPTARG" != "" ]]; then
970             echo "The -r does not take additional parameter. Use the -h for additional info"
971             exit
972         fi
973         ;;
974         h | \?) cat<<EOU
975 Usage: $0 [-a AMI_ID] [-s Security_group_ID] [-k KEY_NAME] [-i instances] [-r] [-p parameters]
976     -a AMI_ID           ID of the AMI to use
977     -s Security_group_ID   ID of the security group to use
978     -k KEY_NAME          Specify the name of the key to use without .pem
979     -i instances          List of instances to start es: "3:m3.medium 2:c4.2xlarge"
980     -p parameters        Parameters to pass to the run-*.*sh script as "... ...."
981     -r                  Enable resume mode
982 EOU
983         exit
984     ;;

```

```

985     esac
986 done
987
988 #Check if the selected AMI is available, if not give option to chose from list of owned
989 .. availabe AMIs
990 while [[ `aws ec2 describe-images --image-id $AMI_ID 2>&1 | grep "\"Name\":"` = "" ]]; do
991     echo "The specified ami does not exist. Do you wish to choose one from the ones you
992 .. own?(y/n)"
993     read_choice 5 "y" "n" "yes" "no"
994     if [[ `echo $CHOICE | grep y` = "" ]]; then
995         exit
996     else
997         IMAGES=`aws ec2 describe-images --filters Name=state,Values=available --owners
998 .. $USER_ID`
999         IMAGES_NAME=`echo "$IMAGES" | grep "\"Name\":" | cut -d "\"" -f 4`
1000        IMAGES_ID=`echo "$IMAGES" | grep "\"ImageId\":" | cut -d "\"" -f 4`
1001        make_list "$IMAGES_NAME"
1002        read_choice 5 `seq $LENGHT`
1003        AMI_ID=`echo "$IMAGES_ID" | head -n $CHOICE | tail -n 1`
1004        #echo $AMI_ID
1005    fi
1006    unset IMAGES
1007    unset IMAGES_NAME
1008    unset IMAGES_ID
1009 done
1010
1011 #Check if the chosen key is available and recognized by aws
1012 while [[ `find $KEYS_PATH -name "$KEY.pem"` = "" || `aws ec2 describe-key-pairs | grep $KEY` =
1013 .. "" ]]; do
1014     echo "You dont own the specified key or it is not recognized by amazon."
1015     echo "Do you wish to choose it from the ones listed in aws that you own as well?(y/n)"
1016     read_choice 5 "y" "n" "yes" "no"
1017     if [[ `echo $CHOICE | grep y` = "" ]]; then
1018         exit
1019     else
1020         KEYS=`find $KEYS_PATH -name "*.pem"`
1021         KEYS_DATA=`aws ec2 describe-key-pairs`
1022         C_COUNT=0

```

```

1023             while read -r line; do
1024                 line=${line::4}
1025                 FIELD=`echo $line | grep "/" -o | wc -l`
1026                 let FIELD++
1027                 KEY=`echo $line | cut -d "/" -f $FIELD`
1028                 if [[ `echo "$KEYS_DATA" | grep "$KEY"` != "" ]]; then
1029                     KEY_LIST[$C_COUNT]="$KEY"
1030                     let C_COUNT++
1031                 fi
1032             done <<< "$KEYS"
1033             if [[ $C_COUNT -lt 1 ]]; then
1034                 echo "No matching key found. Exiting..."
1035                 exit
1036             fi
1037             make_list "${KEY_LIST[@]}"
1038             read_choice 5 `seq $LENGHT`
1039             let CHOICE--
1040             KEY="${KEY_LIST[$CHOICE]}"
1041             #echo $KEY
1042         fi
1043         unset KEY_LIST
1044         unset KEYS
1045         unset KEYS_DATA
1046     done
1047
1048     while [[ `aws ec2 describe-security-groups --group-ids $SG_ID 2>&1 | grep "\"GroupId\":"` = "" ]]
1049 .. ]]; do
1050         echo "The specified security group does not exist.Do you wish to chose one frome those
1051 .. available on aws?(y/n)"
1052         read_choice 5 "y" "n" "yes" "no"
1053         if [[ `echo $CHOICE | grep y` = "" ]]; then
1054             exit
1055         else
1056             SGS=`aws ec2 describe-security-groups`
1057             SGS_NAME=`echo "$SGS" | grep "\"GroupName\":" | cut -d "\"" -f 4`
1058             echo "$SGS_NAME"
1059             SGS_ID=`echo "$SGS" | grep "\"GroupId\":" | cut -d "\"" -f 4`
1060             make_list "$SGS_NAME"

```

```

1055     read_choice 5 `seq $LENGHT`  

1056     SG_ID=`echo "$SGS_ID" | head -n $CHOICE | tail -n 1`  

1057     echo $SG_ID  

1058   fi  

1059   unset SGS  

1060   unset SGS_NAME  

1061   unset SGS_ID  

1062 done  

1063  

1064 mkdir -p error_logs  

1065  

1066 echo "Chose the kind of benchmarks to execute:"  

1067  

1068 make_list "${benchmarks[@]}"  

1069 read_choice 5 `seq $LENGHT`  

1070 let CHOICE--  

1071  

1072 benchmark=${benchmarks[$CHOICE]}  

1073 vol_size=${req_vol_size[$CHOICE]}  

1074 if [[ -z "$PARAM" ]]; then  

1075   PARAM=${parameters[$CHOICE]}  

1076 fi  

1077  

1078 #If a profile is associated with the benchmark, enable it for the run-instances command  

1079 if [[ "${profiles[$CHOICE]}" != "" ]]; then  

1080   profile="--iam-instance-profile Name=\"${profiles[$CHOICE]}\""  

1081 else  

1082   profile=""  

1083 fi  

1084  

1085 #echo "$benchmark:$vol_size:$profile"  

1086 #echo "${instances[@]}"  

1087  

1088 #####  

1089 #####  

1090 #Find an ebs volume with the same name of the benchmark containing the files. If not terminate  

... script

```

```

1091 #This script wont work if all the require files aren't in an ebs volume with the Name tag set  

... as the benchmark name  

1092 #Exception for s3_stresstest_d/u that go with the same volume s3_stresstest  

1093  

1094 if [[ `echo $benchmark | grep s3_stresstest` != "" || "$benchmark" = "test_fermigrid" ]]; then  

1095   a_benchmark="s3_stresstest"  

1096 else  

1097   a_benchmark=$benchmark  

1098 fi  

1099  

1100 BM_VOLUME_DATA=`aws ec2 describe-volumes --filters Name=tag-value,Values=$a_benchmark`  

1101 BM_VOLUME_ID=`echo "$BM_VOLUME_DATA" | grep -F "VolumeId" | cut -d "\"" -f 4`  

1102 AVAL_ZONE=`echo "$BM_VOLUME_DATA" | grep -F "AvailabilityZone" | cut -d "\"" -f 4`  

1103  

1104 if [ "$BM_VOLUME_ID" = "" ]; then  

1105   echo "Can't find the right volume. Quitting..."  

1106   exit  

1107 fi  

1108  

1109 #echo -e "${instances}\n$PARAM\n$benchmark:$vol_size"  

1110 #exit  

1111  

1112 #####  

... #####  

1113 #Create folder to store the VMS data + Resume work option  

1114 COUNT2=0  

1115 RESUME=0  

1116 mkdir -p $TMP_PATH/$benchmark  

1117 if $RESUME_STATE; then  

1118   if [[ -f $TMP_PATH/$benchmark/VMS ]]; then  

1119     echo "I've found previously loaded tmp data. Do you wish to resume work?(y/n)"  

1120     echo "(Works for scripts that got stopped before the volume creation section or  

... requires manual bypass of script parts)"  

1121     read_choice 5 "y" "n" "yes" "no"  

1122   fi  

1123   if [[ "$CHOICE" != "n" ]]; then  

1124     while read -r line; do  

1125       if [[ "$line" != "" ]]; then

```

```

1126         I_TYPES[`echo "$line" | cut -d ":" -f 1`]=`echo "$line" | cut -d ":" -f 2`  

1127         VM_IDS[$COUNT2]=`echo "$line" | cut -d ":" -f 3`  

1128         VM_PUB_DNS[$COUNT2]=`echo "$line" | cut -d ":" -f 4`  

1129         #echo "$COUNT2 ${I_TYPES[$COUNT2]} ${CM_IDS[$COUNT2]}"  

1130         ... ${VM_PUB_DNS[$COUNT2]}"  

1131         let COUNT2++  

1132         let RESUME++  

1133         fi  

1134         done < $TMP_PATH/$benchmark/VMS  

1135     else  

1136         echo "" > $TMP_PATH/$benchmark/VMS  

1137     fi  

1138  

1139 #####  

1140 #####  

1141 #Starting the required VM  

1142 for ((COUNT=0;COUNT<${#instances[@]};COUNT++)); do  

1143     I_NUM=`echo "${instances[$COUNT]}" | cut -d ":" -f 1`  

1144     I_TYPE=`echo "${instances[$COUNT]}" | cut -d ":" -f 2`  

1145  

1146     if [[ $I_NUM -gt $SAME_INSTANCE_LIMIT ]]; then  

1147         echo "The number ($I_NUM) specified for instance $I_TYPE exceeds the limit of  

1148         $SAME_INSTANCE_LIMIT."  

1149         echo "Bringing it down to the maximum allowed..."  

1150         I_NUM=$SAME_INSTANCE_LIMIT  

1151     fi  

1152     echo "Launching $I_NUM $I_TYPE instances"  

1153  

1154     INSTANCE_DATA=`aws ec2 run-instances --count $I_NUM --image-id $AMI_ID --instance-type  

1155     ... $I_TYPE --security-group-ids $SG_ID --placement  

1156     ... AvailabilityZone=$AVAIL_ZONE,GroupName="",Tenancy=default --key-name $KEY $profile`  

1157  

1158     if [[ $RESUME -gt 0 && "${I_TYPES[$(expr $RESUME - 1)]}" = "$I_TYPE" ]]; then  

1159         I_COUNT=$RESUME  

1160     else  

1161         I_COUNT=1

```

```

1159     fi  

1160     L_COUNT=1  

1161     while [[ $I_COUNT -le $I_NUM ]]; do  

1162         I_TYPES[$COUNT2]=$I_TYPE  

1163         INSTANCE_ID=`echo "$INSTANCE_DATA" | grep -F "InstanceId" | head -n $L_COUNT | tail -n  

1164         ... 1 | cut -d "\'" -f 4`  

1165         VM_IDS[$COUNT2]=$INSTANCE_ID  

1166         aws ec2 create-tags --resources $INSTANCE_ID --tags  

1167         ... KeyName,Value=$USER-$I_TYPE-$benchmark-$I_COUNT Key=User,Value=$USER>/dev/null  

1168         let I_COUNT++  

1169         let L_COUNT++  

1170         let COUNT2++  

1171     done  

1172 done  

1173 LAST_INSTANCE=$((expr $COUNT2 - 1))  

1174 #####  

1175 #Get the public DNS after they are available  

1176 echo "Waiting for the DNS..."  

1177 while [[ `aws ec2 describe-instances --instance-ids ${VM_IDS[$LAST_INSTANCE]} | grep -F  

1178     ... "PublicDnsName" | cut -d "\'" -f 4 | uniq` = "" ]]; do  

1179     sleep 10  

1180 done  

1181 sleep 5  

1182 for ((COUNT=$RESUME;COUNT<${#VM_IDS[@]};COUNT++)); do  

1183     INSTANCE_DATA=`aws ec2 describe-instances --instance-ids ${VM_IDS[$COUNT]}`  

1184     PUB_DNS=`echo "$INSTANCE_DATA" | grep -F "PublicDnsName" | cut -d "\'" -f 4 | uniq`  

1185     #Print VMS data in tmp file  

1186     echo "$COUNT:${I_TYPES[$COUNT]}:${VM_IDS[$COUNT]}:$PUB_DNS" >> $TMP_PATH/$benchmark/VMS  

1187     VM_PUB_DNS[$COUNT]=$PUB_DNS  

1188 done  

1189 #####  

1190 #Create extra volumes  

1191 for ((COUNT=0;COUNT<${#VM_IDS[@]};COUNT++)); do

```

```

1192     if [[ "${I_TYPES[$COUNT]}" = "${I_TYPES[$(expr $COUNT - 1)]}" && $COUNT -gt 0 ]]; then
1193         let I_COUNT++
1194     else
1195         I_COUNT=1
1196     fi
1197     echo "Creating extradisk for instance ${I_TYPES[$COUNT]}-$I_COUNT"
1198     VOLUME_DATA=`aws ec2 create-volume --size $vol_size --availability-zone us-west-2a
... --volume-type gp2`
1199     VOLUME_ID[$COUNT]=`echo "$VOLUME_DATA" | grep -F "VolumeId" | cut -d "\"" -f 4`
1200 done
1201 #####
1202 #####Wait until all VM are done initialazing
1203 echo "Waiting for the instances to be initialized..."
1204 for ((COUNT=0;COUNT<${#VM_IDS[@]};COUNT++)); do
1205     while [[ `nmap -p 22 ${VM_PUB_DNS[$COUNT]} | grep open` = "" ]]; do
1206         sleep 15
1207     done
1208 done
1209 #####
1210 #####
1211 #####Disable the self-stop service and register the connection to the DNS (won't ask yes/no
... afterwards)
1212 for ((COUNT=0;COUNT<${#VM_IDS[@]};COUNT++)); do
1213     ssh -o "StrictHostKeyChecking no" -i $KEYS_PATH/$KEY.pem root@$VM_PUB_DNS[$COUNT]
... "service glideinwms-pilot stop">>/dev/null
1214     #echo "Stopped service in instance ${instances[$COUNT]}."
1215 done
1216 #####
1217 #####
1218 #####Waiting for volumes to create
1219 echo "Waiting for volumes to create..."
1220 for ((COUNT=0;COUNT<${#VOLUME_ID[@]};COUNT++)); do

```

```

1224     while [[ `aws ec2 describe-volumes --volume-ids ${VOLUME_ID[$COUNT]} | grep available` =
... "" ]]; do
1225         #echo "Waiting for disk to create..."
1226         sleep 5
1227     done
1228     if [[ "${I_TYPES[$COUNT]}" = "${I_TYPES[$(expr $COUNT - 1)]}" && $COUNT -gt 0 ]]; then
1229         let I_COUNT++
1230     else
1231         I_COUNT=1
1232     fi
1233     aws ec2 create-tags --resources ${VOLUME_ID[$COUNT]} --tags
... Key=$NAME,Value=$USER-${I_TYPES[$COUNT]}-$benchmark-$I_COUNT Key=user,Value=$USER>/dev/null
1234 done
1235 #####
1236 #####Attaching Volumes
1237 echo "Attaching Volumes..."
1238 for ((COUNT=0;COUNT<${#VOLUME_ID[@]};COUNT++)); do
1239     aws ec2 attach-volume --volume-id ${VOLUME_ID[$COUNT]} --instance-id ${VM_IDS[$COUNT]}
... --device /dev/sdf>/dev/null
1240 done
1241 #####
1242 #####
1243 #####Waiting for volumes to attach
1244 echo "Waiting for volumes to attach and making filesystem..."
1245 for ((COUNT=0;COUNT<${#VOLUME_ID[@]};COUNT++)); do
1246     while [[ `aws ec2 describe-volumes --volume-ids ${VOLUME_ID[$COUNT]} | grep attaching` !=
... "" ]]; do
1247         sleep 3
1248     done
1249     if [[ "${I_TYPES[$COUNT]}" = "${I_TYPES[$(expr $COUNT - 1)]}" && $COUNT -gt 0 ]]; then
1250         let I_COUNT++
1251     else
1252         I_COUNT=1
1253     fi
1254     echo "Making file system ext4 for ${I_TYPES[$COUNT]}-$I_COUNT..."

```

```

1256     ssh -i $KEYS_PATH/$KEY.pem root@$VM_PUB_DNS[$COUNT]} "mkfs.ext4 /dev/xvdf
1257     .. 2>/dev/null">/dev/null &
1258 done
1259 :
1260 #Changed to create for all
1261 ######
1262 #If the instance doesn't have a storage volume create it and make the system file as ext4
1263 echo "Creating extrastorage if needed..."
1264 for ((COUNT=0;COUNT<${#VM_IDS[@]};COUNT++)); do
1265     #if [[ `echo "${instances[$COUNT]}" | grep 3` = "" ]]; then
1266     if [[ "${I_TYPES[$COUNT]}" = "${I_TYPES[$(expr $COUNT - 1)]}" && $COUNT -gt 0 ]]; then
1267         let I_COUNT++
1268     else
1269         I_COUNT=1
1270     fi
1271     echo "Creating extradisk for instance ${I_TYPES[$COUNT]}-$I_COUNT"
1272     VOLUME_DATA=`aws ec2 create-volume --size $vol_size --availability-zone us-west-2a
1273     --volume-type gp2`
1274     VOLUME_ID=`echo "$VOLUME_DATA" | grep -F "VolumeId" | cut -d "\\" -f 4` 
1275     aws ec2 create-tags --resources $VOLUME_ID --tags
1276     Key=Name,Value=$USER-$instances[$COUNT]-$benchmark-$I_COUNT Key=user,Value=$USER>/dev/null
1277     while [[ `aws ec2 describe-volumes --volume-ids $VOLUME_ID | grep available` = "" ]]; do
1278         #echo "Waiting for disk to create..."
1279         sleep 5
1280     done
1281     aws ec2 attach-volume --volume-id $VOLUME_ID --instance-id ${VM_IDS[$COUNT]} --device
1282     /dev/sdf>/dev/null
1283     while [[ `aws ec2 describe-volumes --volume-ids $VOLUME_ID | grep attaching` != "" ]]; do
1284         sleep 3
1285     done
1286     ssh -i $KEYS_PATH/$KEY.pem root@$VM_PUB_DNS[$COUNT]} "mkfs.ext4 /dev/xvdf
1287     .. 2>/dev/null">/dev/null
1288     #echo "Volume attaccato a ${instances[$COUNT]}"
1289     #else
1290     #echo "No extradisk for instance ${instances[$COUNT]}."
1291     #fi

```

```

1288 done'
1289 #####
1290 #####
1291 #Mount the volumes and transfer the benchmark via script
1292 echo "Mounting volumes and transferring benchmarks..."
1293 : 'cat <<TRANSFER > transfer.sh
1294 #!/bin/bash
1295
1296 mkdir -p /scratch
1297 mkdir -p /scratch02
1298
1299 if [[ `fdisk -l | grep xvdb` != "" ]]; then
1300     mount /dev/xvdb /scratch
1301     mount /dev/xvdf /scratch02
1302 else
1303     mount /dev/xvdf /scratch/
1304     mount /dev/xvdg /scratch02/
1305 fi
1306 cp /scratch02/* /scratch/
1307 if [[ `fdisk -l | grep xvdb` != "" ]]; then
1308     umount /dev/xvdf
1309 else
1310     umount /dev/xvdg
1311 fi
1312 rm -r -f /scratch02/
1313 exit
1314 TRANSFER'
1315
1316 cat <<TRANSFER > transfer.sh
1317 #!/bin/bash
1318
1319 mkdir -p /scratch
1320 mkdir -p /scratch02
1321
1322 mount /dev/xvdf /scratch
1323 mount /dev/xvdg /scratch02
1324

```

```

1325 cp /scratch02/* /scratch/
1326 umount /dev/xvdg
1328
1329 rm -r -f /scratch02/
1330 exit
1331 TRANSFER
1332
1333 for ((COUNT=0;COUNT<${#VM_IDS[@]};COUNT++)); do
1334     scp -i $KEYS_PATH/$KEY.pem transfer.sh root@$VM_PUB_DNS[$COUNT]:/root/>/dev/null 2>&1
1335     : 'if [[ `echo "${instances[$COUNT]}" | grep 3` = "" ]]; then
1336         aws ec2 attach-volume --volume-id $BM_VOLUME_ID --instance-id ${VM_IDS[$COUNT]}
1337         --device /dev/sdg>/dev/null
1338     else
1339         aws ec2 attach-volume --volume-id $BM_VOLUME_ID --instance-id ${VM_IDS[$COUNT]}
1340         --device /dev/sdf>/dev/null
1341     fi'
1342
1343     aws ec2 attach-volume --volume-id $BM_VOLUME_ID --instance-id ${VM_IDS[$COUNT]} --device
1344     /dev/sdg>/dev/null
1345     while [[ `aws ec2 describe-volumes --volume-ids $BM_VOLUME_ID | grep attaching` != "" ]];
1346     do
1347         sleep 3
1348     done
1349     ssh -i $KEYS_PATH/$KEY.pem root@$VM_PUB_DNS[$COUNT] "sh /root/transfer.sh">/dev/null
1350     2>>error_logs/transfer_error-$COUNT.log
1351     ssh -i $KEYS_PATH/$KEY.pem root@$VM_PUB_DNS[$COUNT] "rm /root/transfer.sh">/dev/null
1352     2>&1
1353     aws ec2 detach-volume --volume-id $BM_VOLUME_ID --instance-id ${VM_IDS[$COUNT]} --device
1354     /dev/sdg>/dev/null
1355     : 'if [[ `echo "${instances[$COUNT]}" | grep 3` = "" ]]; then
1356         aws ec2 detach-volume --volume-id $BM_VOLUME_ID --instance-id ${VM_IDS[$COUNT]}
1357         --device /dev/sdg>/dev/null
1358     else
1359         aws ec2 detach-volume --volume-id $BM_VOLUME_ID --instance-id ${VM_IDS[$COUNT]}
1360         --device /dev/sdf>/dev/null
1361     fi'

```

```

1354     while [[ `aws ec2 describe-volumes --volume-ids $BM_VOLUME_ID | grep detaching` != "" ]];
1355     do
1356         sleep 5
1357     done
1358 done
1359 rm -f transfer.sh
1360
1361 #####
1362 ##### Start the benchmark
1363 #Start the benchmark
1364 echo "Starting the benchmark on every instance..."
1365 for ((COUNT=0;COUNT<${#VM_IDS[@]};COUNT++)); do
1366     scp -i $KEYS_PATH/$KEY.pem $FILES_PATH/run-$benchmark.sh
1367     root@$VM_PUB_DNS[$COUNT]:/root/>/dev/null 2>&1
1368     ssh -i $KEYS_PATH/$KEY.pem root@$VM_PUB_DNS[$COUNT] "chmod 755 /root/run-$benchmark.sh"
1369     >/dev/null 2>&1
1370     ssh -i $KEYS_PATH/$KEY.pem root@$VM_PUB_DNS[$COUNT] "/root/run-$benchmark.sh
1371     $PARAM">/dev/null 2>>error_logs/run_error_$COUNT.log &
1372 done
1373
1374 echo "I'm done. Wait for the benchmark to be over and than launch the crop_results.sh script."
1375
1376
1377 -----
1378 -----
1379 -----
1380
1381 File: predictiondecisionengine/trunk/bin/Fermilab/From_EBS/run-hepspec06.sh
1382 #!/bin/bash
1383
1384 if [[ `which rpm` = "" || `which rpm | grep -F "no aws"` != "" ]]; then
1385     yum install rpm

```

```

1387  fi
1388  if [[ `rpm -qa gcc-c++` = "" ]]; then
1389      yum install gcc-c++
1390  fi
1391
1392  cd /scratch
1393  tar xvzf SPEC_CPU2006v1.1.tar.gz
1394
1395  cd SPEC_CPU2006v1.1
1396
1397  ln -s /usr/bin/gcc /usr/local/bin/gcc
1398  ./install.sh -d ../install/ -f
1399
1400  cd ..
1401  mkdir hepspec
1402  cd hepspec
1403  tar xvzf ../spec2k6-2.23.tar.gz
1404  cp linux64-gcc_cern.cfg ..../install/config
1405
1406
1407
1408  cd ../install
1409  . ./shrc
1410  #runspec --config=linux64-gcc_cern.cfg all_cpp
1411  mkdir -p ./results
1412  #mv ./result ./results/Single_core
1413
1414  COUNT=`grep -c "processor" /proc/cpuinfo`;
1415  for i in `seq $COUNT`;
1416  do
1417      runspec --config=linux64-gcc_cern.cfg all_cpp &
1418  done
1419  wait
1420
1421  mv ./result ./results/All_cores
1422
1423
1424 -----

```

```

1425 -----
1426 -----
1427
1428
1429 File: predictiondecisionengine/trunk/bin/Fermilab/From_EBS/run-s3_stresstest_d.sh
1430 #!/bin/bash
1431
1432 TEST_BUCKET=grassano-test
1433 TEST_FILE=ACEE623F-B1A8-E411-8672-0025905938B4.root
1434 : 'service glideinwms-pilot stop'
1435
1436 mkfs.ext4 /dev/xvdf 2>/dev/null
1437 mkdir -p /scratch
1438 mount /dev/xvdf /scratch'
1439
1440 cd /scratch
1441
1442 if [[ `which aws` = "" || `which aws | grep -F "no aws"` != "" ]]; then
1443     #wget https://s3.amazonaws.com/aws-cli/awscli-bundle.zip
1444     unzip awscli-bundle.zip
1445     python awscli-bundle/install
1446     PATH=$PATH:/root/.local/lib/aws/bin
1447     export PATH
1448     #mkdir /root/.aws
1449     #mv /scratch/config /root/.aws/
1450     #chmod 400 /root/.aws/config
1451 fi
1452
1453
1454 echo "Download test" > s3_stress.log
1455 LIST="1 10 100"
1456 if [[ $# -gt 0 ]]; then
1457     LIST="$@"
1458 fi
1459 #NUM=1
1460 for NUM in $LIST; do
1461     mkdir -p $NUM
1462     echo "Simultaneous download of $NUM copies..." >>s3_stress.log

```

```

1463     date >>s3_stress.log
1464     for n in `seq $NUM`; do
1465         aws s3 cp s3://$TEST_BUCKET/$TEST_FILE $NUM/$n.root > $NUM/$n.log 2>&1 &
1466     done
1467     wait
1468     date >>s3_stress.log
1469     echo -e "`ls -l $NUM/ | grep 1073604531 | wc -l` out of $NUM successfully downloaded\n"
... >>s3_stress.log
1470     #let NUM*=10
1471     #echo $NUM
1472 done
1473
1474 exit
1475
1476 -----
1477 -----
1478 -----
1479
1480
1481 File: predictiondecisionengine/trunk/bin/Fermilab/From_EBS/run-s3_stresstest_u.sh
1482 #!/bin/bash
1483
1484 TEST_BUCKET=grassano-test
1485 TEST_FILE=ACEE623F-B1A8-E411-8672-0025905938B4.root
1486 : 'service glideinwms-pilot stop'
1487
1488 mkfs.ext4 /dev/xvdf 2>/dev/null
1489 mkdir -p /scratch
1490 mount /dev/xvdf /scratch'
1491
1492 cd /scratch
1493
1494 if [[ `which aws` = "" || `which aws | grep -F "no aws"` != "" ]]; then
1495     #wget https://s3.amazonaws.com/aws-cli/awscli-bundle.zip
1496     unzip awscli-bundle.zip
1497     python awscli-bundle/install
1498     PATH=$PATH:/root/.local/lib/aws/bin
1499     export PATH

```

```

1500     #mkdir /root/.aws
1501     #mv /scratch/config /root/.aws/
1502     #chmod 400 /root/.aws/config
1503 fi
1504
1505 ID=$RANDOM
1506 echo "Upload test ID=$ID" > s3_stress.log
1507 LIST="1 10 100"
1508 if [[ $# -gt 0 ]]; then
1509     LIST="$@"
1510 fi
1511 #NUM=1
1512 for NUM in $LIST; do
1513     mkdir -p $NUM
1514     echo "Simultaneous upload of $NUM copies..." >>s3_stress.log
1515     date >>s3_stress.log
1516     for n in `seq $NUM`; do
1517         aws s3 cp $TEST_FILE s3://$TEST_BUCKET/$NUM-$ID/$n.root > $NUM/$n.log 2>&1 &
1518     done
1519     wait
1520     date >>s3_stress.log
1521     echo -e "`aws s3api list-objects --bucket $TEST_BUCKET --prefix $NUM-$ID/ | grep
... 1073604531 | wc -l` out of $NUM successfully uploaded\n" >>s3_stress.log
1522     #let NUM*=10
1523     #echo $NUM
1524 done
1525
1526 exit
1527
1528 -----
1529 -----
1530 -----
1531
1532
1533 File: predictiondecisionengine/trunk/bin/Fermilab/From_EBS/run-test_fermigrid.sh
1534 #!/bin/bash
1535
1536 TEST_BUCKET=grassano-test

```

```

1537 TEST_FILE=ACEE623F-B1A8-E411-8672-0025905938B4.root
1538 cd /scratch
1539 mv x509up_u50685 /tmp/x509up_u0
1540 voms-proxy-init -noregen -voms fermilab:/fermilab
1541 :
1542 if [[ `which aws` = "" || `which aws | grep -F "no aws"` != "" ]]; then
1543     wget https://s3.amazonaws.com/aws-cli/awscli-bundle.zip
1544     unzip awscli-bundle.zip
1545     python awscli-bundle/install
1546     PATH=$PATH:/root/.local/lib/aws/bin
1547     export PATH
1548     #mkdir /root/.aws
1549     #mv /scratch/config /root/.aws/
1550     #chmod 400 /root/.aws/config
1551 fi
1552 '
1553
1554 DEST=gsiftp://fnlca1.fnal.gov:2811/fermigrid202/grassano
1555 ID=$RANDOM
1556 echo "Upload test ID=$ID" > fermigrid_upload.log
1557 echo "" > data.txt
1558 if [[ ! -f $TEST_FILE ]]; then
1559     aws s3 cp s3://$TEST_BUCKET/$TEST_FILE .
1560 fi
1561 LIST="1 5 10 20"
1562 if [[ $# -gt 0 ]]; then
1563     LIST="$@"
1564 fi
1565 for NUM in $LIST; do
1566     #mkdir -p $NUM
1567     echo "Simultaneous upload of $NUM copies..." >> fermigrid_upload.log
1568     echo "" > data.txt
1569     for n in `seq $NUM`; do
1570         echo "file:///scratch/$TEST_FILE $DEST/$ID/$n.root" >> data.txt
1571     done
1572     date >> fermigrid_upload.log
1573     globus-url-copy -cd -fast -p 4 -cc $NUM -f data.txt >> globus.log 2>&1
1574

```

```

1575     date >> fermigrid_upload.log
1576     echo -e "`edg-gridftp-ls -v $DEST/$ID/ | grep 1073604531 | wc -l` out of $NUM successfully
... uploaded\n" >> fermigrid_upload.log
1577     for n in `seq $NUM`; do
1578         edg-gridftp-rm $DEST/$ID/$n.root
1579     done
1580     edg-gridftp-rmdir $DEST/$ID/
1581     #let NUM*=10
1582     #echo $NUM
1583 done
1584
1585 exit
1586 -----
1587 -----
1588 -----
1589 -----
1590
1591
1592 File: predictiondecisionengine/trunk/bin/Fermilab/From_EBS/run-tt_bar_gensim.sh
1593#!/bin/bash
1594
1595 cd /scratch
1596
1597 tar -zxf benchmark-2015.tgz
1598
1599 groupadd cms_admin
1600 useradd -g cms_admin cms_admin
1601
1602 ln -s /scratch/cms/ /tmp/cms
1603 chown -R cms_admin.cms_admin /tmp/cms/
1604
1605 cd /tmp/cms/
1606 su cms_admin -c "sh benchmark-2015-gensim.sh 1">log
1607 mv results results.Single_core
1608
1609 su cms_admin -c "sh benchmark-2015-gensim.sh">log
1610 mv results results.All_cores
1611

```

```

1612 -----
1613 -----
1614 -----
1615 
1616 File: predictiondecisionengine/trunk/bin/Fermilab/From_EBS/run_tt_bar_reco.sh
1617 #!/bin/bash
1618
1619 groupadd cms_admin
1620 useradd -g cms_admin cms_admin
1621
1622 cd /scratch
1623 tar -zxf /scratch/benchmark-2015.tgz
1625 ln -s /scratch/tmp/cms/ /tmp/cms
1626
1627 mv /scratch/ACEE623F-B1A8-E411-8672-0025905938B4.root /tmp/cms
1628 mv /scratch/ttbarGENSIM.root /tmp/cms
1629 mv /scratch/step2-50ns-4x-6400.root /tmp/cms
1630 mv -f /scratch/*.sh /tmp/cms
1631
1632 mv /scratch/frontier-cache.tgz /tmp/cms
1633 cd /tmp/cms/
1634 tar -zxf /tmp/cms/frontier-cache.tgz
1635
1636 chown -R cms_admin:cms_admin /tmp/cms/
1637
1638 su cms_admin -c "sh start_x_benchmarks_reco.sh">>>log &
1639
1640 exit
1641
1642 -----
1643 -----
1644 -----
1645 
1646
1647 File: predictiondecisionengine/trunk/bin/Fermilab/run-hepspec06.sh
1648 #!/bin/bash
1649

```

```

1650 shift
1651 shift
1652
1653 TEST_BUCKET=grassano-test
1654
1655 mkdir -p /scratch/
1656 mount /dev/xvdf /scratch/
1657
1658 if [[ `which rpm` = "" || `which rpm | grep -F "no aws" ` != "" ]]; then
1659     yum install rpm
1660 fi
1661
1662 if [[ `rpm -qa gcc-c++` = "" ]]; then
1663     yum install gcc-c++
1664 fi
1665
1666 cd /scratch
1667
1668 if [[ `which aws` = "" || `which aws | grep -F "no aws" ` != "" ]]; then
1669     if [[ -e /root/.local/lib/aws/bin ]]; then
1670         export PATH=$PATH:/root/.local/lib/aws/bin
1671     else
1672         wget https://s3.amazonaws.com/aws-cli/awscli-bundle.zip
1673         unzip awscli-bundle.zip
1674         python awscli-bundle/install
1675         export PATH=$PATH:/root/.local/lib/aws/bin
1676         #mkdir /root/.aws
1677         #mv /scratch/config /root/.aws/
1678         #chmod 400 /root/.aws/config
1679     fi
1680 fi
1681
1682 aws s3 cp --recursive s3://$TEST_BUCKET/hepspec06/ .
1683
1684 tar xvzf SPEC_CPU2006v1.1.tar.gz
1685
1686 cd SPEC_CPU2006v1.1
1687

```

```

1688 ln -s /usr/bin/gcc /usr/local/bin/gcc
1689 ./install.sh -d .../install/ -f
1690
1691 cd ..
1692 mkdir hepspec
1693 cd hepspec
1694 tar xvzf ../spec2k6-2.23.tar.gz
1695 cp linux64-gcc_cern.cfg .../install/config
1696
1697
1698 cd ..../install
1699 . ./shrc
1700 if [[ $# -eq 1 && $1 -eq 0 ]]; then
1701     exit
1702 fi
1703 runspec --config=linux64-gcc_cern.cfg all_cpp
1704 mkdir -p ./results
1705 mv ./result ./results/Single_core
1706
1707 COUNT=`grep -c "^processor" /proc/cpuinfo`;
1708 for i in `seq $COUNT`;
1709 do
1710     runspec --config=linux64-gcc_cern.cfg all_cpp &
1711 done
1712 wait
1713
1714 mv ./result ./results/All_cores
1715
1716
1717 -----
1718 -----
1719 -----
1720
1721
1722 File: predictiondecisionengine/trunk/bin/Fermilab/run-s3_stresstest_d.sh
1723 #!/bin/bash
1724
1725 shift

```

```

1726 shift
1727 TEST_BUCKET=grassano-test
1728 TEST_FILE=ACEE623F-B1A8-E411-8672-0025905938B4.root
1729
1730 mkdir -p /scratch/
1731 mount /dev/xvdf /scratch/
1732 cd /scratch
1733
1734 if [[ `which aws` = "" || `which aws | grep -F "no aws"` != "" ]]; then
1735     if [[ -e /root/.local/lib/aws/bin ]]; then
1736         export PATH=$PATH:/root/.local/lib/aws/bin
1737     else
1738         wget https://s3.amazonaws.com/aws-cli/awscli-bundle.zip
1739         unzip awscli-bundle.zip
1740         python awscli-bundle/install
1741         export PATH=$PATH:/root/.local/lib/aws/bin
1742         #mkdir /root/.aws
1743         #mv /scratch/config /root/.aws/
1744         #chmod 400 /root/.aws/config
1745     fi
1746 fi
1747
1748 echo "Download test" > s3_stress.log
1749 LIST="1 10 100"
1750 if [[ $# -gt 0 ]]; then
1751     LIST="$@"
1752 fi
1753 if [[ $# -eq 1 && $1 -eq 0 ]]; then
1754     exit
1755 fi
1756 #NUM=1
1757 for NUM in $LIST; do
1758     mkdir -p $NUM
1759     echo "Simultaneous download of $NUM copies..." >>s3_stress.log
1760     date >>s3_stress.log
1761     for n in `seq $NUM`; do
1762         aws s3 cp s3://$TEST_BUCKET/s3_stresstest/$TEST_FILE $NUM/$n.root > $NUM/$n.log 2>&1 &
1763     done

```

```

1764     wait
1765     date >>s3_stress.log
1766     echo -e "`ls -l $NUM/ | grep 1073604531 | wc -l` out of $NUM successfully downloaded\n"
...>>s3_stress.log
1767     #let NUM*=10
1768     #echo $NUM
1769 done
1770
1771 exit
1772 -----
1773 -----
1774 -----
1775 -----
1776
1777 File: predictiondecisionengine/trunk/bin/Fermilab/run-s3_stresstest_u.sh
1779 #!/bin/bash
1780
1781 TIME=$1
1782 shift
1783 ID=$1
1784 shift
1785 sleep $((240 + `date -d "$TIME" +%s` - `date +%s`)) &
1786 WAIT_ID=$!
1787
1788 TEST_BUCKET=grassano-test
1789 TEST_FILE=ACEE623F-B1A8-E411-8672-0025905938B4.root
1790
1791 mkdir -p /scratch/
1792 mount /dev/xvdf /scratch/
1793 cd /scratch
1794
1795 if [[ `which aws` = "" || `which aws | grep -F "no aws"` != "" ]]; then
1796     if [[ -e /root/.local/lib/aws/bin ]]; then
1797         export PATH=$PATH:/root/.local/lib/aws/bin
1798     else
1799         wget https://s3.amazonaws.com/aws-cli/awscli-bundle.zip
1800         unzip awscli-bundle.zip

```

```

1801     python awscli-bundle/install
1802     export PATH=$PATH:/root/.local/lib/aws/bin
1803     #mkdir /root/.aws
1804     #mv /scratch/config /root/.aws/
1805     #chmod 400 /root/.aws/config
1806     fi
1807 fi
1808
1809 aws s3 cp s3://$TEST_BUCKET/s3_stresstest/$TEST_FILE .
1810
1811 #ID=$RANDOM
1812 echo "Upload test ID=$ID" > s3_stress.log
1813 LIST="1 10 100"
1814 if [[ $# -gt 0 ]]; then
1815     LIST="$@"
1816 fi
1817 if [[ $# -eq 1 && $1 -eq 0 ]]; then
1818     exit
1819 fi
1820
1821 wait $WAIT_ID
1822 for NUM in $LIST; do
1823     mkdir -p $NUM
1824     echo "Simultaneous upload of $NUM copies..." >>s3_stress.log
1825     date >>s3_stress.log
1826     for n in `seq $NUM`; do
1827         aws s3 cp $TEST_FILE s3://$TEST_BUCKET/$NUM-$ID/$n.root > $NUM/$n.log 2>&1 &
1828     done
1829     wait
1830     date >>s3_stress.log
1831     echo -e "`aws s3api list-objects --bucket $TEST_BUCKET --prefix $NUM-$ID/ | grep
... 1073604531 | wc -l` out of $NUM successfully uploaded\n" >>s3_stress.log
1832     #let NUM*=10
1833     #echo $NUM
1834 done
1835
1836 exit
1837

```

```

1838 -----
1839 -----
1840 -----
1841
1842 File: predictiondecisionengine/trunk/bin/Fermilab/run-test_fermigrid.sh
1843 #!/bin/bash
1845
1846 TIME=$1
1847 shift
1848 ID=$1
1849 shift
1850 sleep $((240 + `date -d "$TIME" +%s` - `date +%s`)) &
1851 WAIT_ID=$!
1852
1853 TEST_BUCKET=grassano-test
1854 TEST_FILE=ACEE623F-B1A8-E411-8672-0025905938B4.root
1855
1856 mkdir -p /scratch/
1857 mount /dev/xvdf /scratch/
1858 cd /scratch
1859
1860 if [[ `which aws` = "" || `which aws | grep -F "no aws"` != "" ]]; then
1861     if [[ -e /root/.local/lib/aws/bin ]]; then
1862         export PATH=$PATH:/root/.local/lib/aws/bin
1863     else
1864         wget https://s3.amazonaws.com/aws-cli/awscli-bundle.zip
1865         unzip awscli-bundle.zip
1866         python awscli-bundle/install
1867         export PATH=$PATH:/root/.local/lib/aws/bin
1868         #mkdir /root/.aws
1869         #mv /scratch/config /root/.aws/
1870         #chmod 400 /root/.aws/config
1871     fi
1872 fi
1873
1874 aws s3 cp --recursive s3://$TEST_BUCKET/s3_stresstest/ .
1875

```

```

1876 #Remember to keep the certificates valid and modify the script if the users number changes
1877 # (The 0 is always for root)
1878 mv x509up_u50685 /tmp/x509up_u0
1879 chmod 400 /tmp/x509up_u0
1880 rm -f /etc/grid-security/certificates/*.r0
1881 voms-proxy-init -noregen -voms fermilab:/fermilab
1882
1883 DEST=gsiftp://fnndca1.fnal.gov:2811/fermigrid202/grassano
1884 #ID=$RANDOM
1885 echo "Upload test ID=$ID" > fermigrid_upload.log
1886 #wait $WAIT_ID
1887 echo "" > data.txt
1888 LIST="1 5 10 20"
1889 if [[ $# -gt 0 ]]; then
1890     LIST="$@"
1891 fi
1892 if [[ $# -eq 1 && $1 -eq 0 ]]; then
1893     kill $WAIT_ID
1894     exit
1895 fi
1896 for NUM in $LIST; do
1897     #mkdir -p $NUM
1898     echo "Simultaneous upload of $NUM copies..." >> fermigrid_upload.log
1899     echo "" > data.txt
1900     for n in `seq $NUM`; do
1901         echo "file:///scratch/$TEST_FILE $DEST/$ID/$n.root" >> data.txt
1902     done
1903     date >> fermigrid_upload.log
1904     if [[ $NUM -le 5 ]]; then
1905         CONCURRENCY=$NUM
1906     else
1907         CONCURRENCY=5
1908     fi
1909     globus-url-copy -cd -fast -p 4 -cc $CONCURRENCY -v -f data.txt >> command.log 2>&1
1910     date >> fermigrid_upload.log
1911     echo -e "`edg-gridftp-ls -v $DEST/$ID/ | grep 1073604531 | wc -l` out of $NUM successfully
1912     .. uploaded\n" >> fermigrid_upload.log

```

```

1912     for n in `seq $NUM` ; do
1913         edg-gridftp-rm $DEST/$ID/$n.root
1914     done
1915     edg-gridftp-rmdir $DEST/$ID/
1916     #let NUM*=10
1917     #echo $NUM
1918 done
1919
1920 exit
1921 -----
1922 -----
1923 -----
1924 -----
1925
1926
1927 File: predictiondecisionengine/trunk/bin/Fermilab/run-test_fermigrid2.sh
1928 #!/bin/bash
1929
1930 TIME=$1
1931 shift
1932 ID=$1
1933 shift
1934 sleep $((180 + `date -d "$TIME" +%s` - `date +%s`)) &
1935 WAIT_ID=$!
1936
1937 TEST_BUCKET=grassano-test
1938 TEST_FILE=ACEE623F-B1A8-E411-8672-0025905938B4.root
1939
1940 mkdir -p /scratch/
1941 mount /dev/xvdf /scratch/ >/dev/null 2>&1
1942 cd /scratch
1943
1944 if [[ `which aws` = "" || `which aws | grep -F "no aws"` != "" ]]; then
1945     if [[ -e /root/.local/lib/aws/bin ]]; then
1946         export PATH=$PATH:/root/.local/lib/aws/bin
1947     else
1948         wget https://s3.amazonaws.com/aws-cli/awscli-bundle.zip
1949         unzip awscli-bundle.zip

```

```

1950     python awscli-bundle/install
1951     export PATH=$PATH:/root/.local/lib/aws/bin
1952     #mkdir /root/.aws
1953     #mv /scratch/config /root/.aws/
1954     #chmod 400 /root/.aws/config
1955     fi
1956 fi
1957
1958 aws s3 cp --recursive s3://$TEST_BUCKET/s3_stresstest/ .
1959
1960 #Remember to keep the certificates valid and modify the script if the users number changes
1961 # (The 0 is always for root)
1962 cp x509up_u2904 /tmp/x509up_u0
1963 chmod 400 /tmp/x509up_u0
1964 mv x509up_u2904 /tmp/x509up_u2904
1965 chmod 400 /tmp/x509up_u2904
1966 export X509_USER_CERT=/tmp/x509up_u2904
1967 export X509_USER_KEY=/tmp/x509up_u2904
1968 rm -f /etc/grid-security/certificates/*.r0
1969 voms-proxy-init -noregen -voms fermilab:/fermilab
1970
1971 DEST_XRD=root://cmseos.fnal.gov//eos/uscms/store/user/timm
1972 DEST_SRM=srm://cmseos.fnal.gov:8443/srm/v2/server?SFN=/eos/uscms/store/user/timm
1973 #ID=$RANDOM
1974 echo "Upload test ID=$ID" > fermigrid_upload.log
1975 LIST="1 5 10 20"
1976 if [[ $# -gt 0 ]]; then
1977     LIST="$@"
1978 fi
1979 if [[ $# -eq 1 && $1 -eq 0 ]]; then
1980     kill $WAIT_ID
1981     exit
1982 fi
1983 srmmkdir $DEST_SRM/$ID
1984 wait $WAIT_ID
1985 for NUM in $LIST; do
1986     echo "Simultaneous upload of $NUM copies..." >> fermigrid_upload.log

```

```

1987     date >>fermigrid_upload.log
1988     for n in `seq $NUM` ; do
1989         #srmcp -streams_num=5 file:///$TEST_FILE $DEST_SRM/$ID/$n.root >> command.log 2>&1 &
1990         xrdcp -f -v -N -S 4 /scratch/$TEST_FILE $DEST_XRD/$ID/$n.root >> command.log 2>&1 &
1991     done
1992     wait
1993     date >>fermigrid_upload.log
1994
1995     echo -e "`srmls $DEST_SRM/$ID/ | grep -c 1073604531` out of $NUM successfully uploaded\n"
1996     ->>fermigrid_upload.log
1997     for n in `seq $NUM` ; do
1998         srmrmm $DEST_SRM/$ID/$n.root &
1999     done
2000     wait
2000 done
2001 srmrmdir $DEST_SRM/$ID/
2002
2003 exit
2004
2005 -----
2006 -----
2007 -----
2008
2009
2010 File: predictiondecisionengine/trunk/bin/Fermilab/run-tt_bar_gensim.sh
2011 #!/bin/bash
2012
2013 Shift
2014 shift
2015
2016 mkdir -p /scratch/
2017 mount /dev/xvdf /scratch/
2018 cd /scratch
2019
2020 if [[ `which aws` = "" || `which aws | grep -F "no aws"` != "" ]]; then
2021     if [[ -e /root/.local/lib/aws/bin ]]; then
2022         export PATH=$PATH:/root/.local/lib/aws/bin
2023     else

```

```

2024     wget https://s3.amazonaws.com/aws-cli/awscli-bundle.zip
2025     unzip awscli-bundle.zip
2026     python awscli-bundle/install
2027     export PATH=$PATH:/root/.local/lib/aws/bin
2028     #mkdir /root/.aws
2029     #mv /scratch/config /root/.aws/
2030     #chmod 400 /root/.aws/config
2031     fi
2032 fi
2033
2034 aws s3 cp --recursive s3://$TEST_BUCKET/tt_bar_gensim/ .
2035
2036
2037 tar -zxf benchmark-2015.tgz
2038
2039 groupadd cms_admin
2040 useradd -g cms_admin cms_admin
2041
2042 ln -s /scratch/cms/ /tmp/cms
2043 chown -R cms_admin.cms_admin /tmp/cms/
2044
2045 if [[ $# -eq 1 && $1 -eq 0 ]]; then
2046     exit
2047 fi
2048 cd /tmp/cms/
2049 su cms_admin -c "sh benchmark-2015-gensim.sh 1">log
2050 mv results results.Single_core
2051
2052 su cms_admin -c "sh benchmark-2015-gensim.sh">log
2053 mv results results.All_cores
2054
2055 -----
2056 -----
2057 -----
2058
2059
2060 File: predictiondecisionengine/trunk/bin/Fermilab/run-tt_bar_reco.sh
2061 #!/bin/bash

```

```

2062
2063 groupadd cms_admin
2064 useradd -g cms_admin cms_admin
2065
2066 mkdir -p /scratch/
2067 mount /dev/xvdf /scratch/
2068 cd /scratch
2069
2070 if [[ `which aws` = "" || `which aws | grep -F "no aws"` != "" ]]; then
2071     if [[ -e /root/.local/lib/aws/bin ]]; then
2072         export PATH=$PATH:/root/.local/lib/aws/bin
2073     else
2074         wget https://s3.amazonaws.com/aws-cli/awscli-bundle.zip
2075         unzip awscli-bundle.zip
2076         python awscli-bundle/install
2077         export PATH=$PATH:/root/.local/lib/aws/bin
2078         #mkdir /root/.aws
2079         #mv /scratch/config /root/.aws/
2080         #chmod 400 /root/.aws/config
2081     fi
2082 fi
2083
2084 aws s3 cp s3://$TEST_BUCKET/tt_bar_reco/* ./
2085
2086 tar -zxf /scratch/benchmark-2015.tgz
2087 ln -s /scratch/tmp/cms/ /tmp/cms
2088
2089 mv /scratch/ACEE623F-B1A8-E411-8672-0025905938B4.root /tmp/cms
2090 mv /scratch/ttbarGENSIM.root /tmp/cms
2091 mv /scratch/step2-50ns-4x-6400.root /tmp/cms
2092 mv -f /scratch/*.sh /tmp/cms
2093
2094 mv /scratch/frontier-cache.tgz /tmp/cms
2095 cd /tmp/cms/
2096 tar -zxf /tmp/cms/frontier-cache.tgz
2097
2098 chown -R cms_admin.cms_admin /tmp/cms/
2099

```

```

2100 su cms_admin -c "sh start_x_benchmarks_reco.sh">>>log &
2101
2102 exit
2103
2104 -----
2105 -----
2106 -----
2107
2108
2109 File: predictiondecisionengine/trunk/bin/Fermilab/runparallel.sh
2110 #!/bin/bash
2111
2112 cd ./hepspec2006/install
2113 mkdir ./results
2114 mv ./result ./results/Single_core
2115
2116 ./shrc
2117 COUNT=`grep -c "^processor" /proc/cpuinfo`;
2118 for i in `seq $COUNT`; do
2119     runspec --config=linux64-gcc_cern.cfg all_cpp &
2120 done
2121 wait
2122
2123 mv ./result ./results/All_cores
2124
2125 -----
2126 -----
2127 -----
2128
2129
2130 File: predictiondecisionengine/trunk/bin/functions.sh
2131 #Description: Read the choice to a query and confront it with a list of correct answer
2132 ... limiting the numbers of checks
2133 #Parameters: 1st parameter: number of attempts possible other parameter:List of correct
2134 ... answers
2135 #Usage example (combined with make_list): make_list ...    read_choice 3 `seq $LENGTH`
2136 function read_choice()
2137 {

```

```

2136     local NUM=$1
2137     shift
2138     CHOICES=("${![@]}")
2139     while "true"; do
2140         read CHOICE
2141         CHOICE=`echo $CHOICE | tr YESNO yesno` 
2142         for CMD in ${CHOICES[@]}; do
2143             if [[ "$CHOICE" == "$CMD" ]]; then
2144                 return 0
2145             fi
2146         done
2147         echo "Choice is out of range. Enter again :"
2148         let C_COUNT++
2149         if [[ $C_COUNT -ge $NUM ]]; then
2150             echo "$NUM invalid choices. Exiting..." 
2151             exit
2152         fi
2153     done
2154 }
2155
2156 #Description: Make a numbered list out of an array or list of elements and return the number
2157 #of elements in LENGTH
2158 #Parameters: array or list of elements
2159 #Usage example: make_list "$LIST"      or      make_list "${array[@]}"
2160 function make_list()
2161 {
2162     local C_COUNT=0
2163     if [[ `echo "$1" | wc -l` -gt 1 ]]; then
2164         while read -r line; do
2165             let C_COUNT++
2166             echo -e "$C_COUNT-\t$line"
2167         done <<< "$1"
2168     else
2169         local array=("$@")
2170         for CMD in ${array[@]}; do
2171             let C_COUNT++
2172             echo -e "$C_COUNT-\t$CMD"
2173         done
2174     LENGTH=$C_COUNT
2175 }
2176
2177 #Description: Transform a list of elements written on different lines into an array
2178 #Parameters: list of element
2179 #Usage example: array=($(list_to_array "$LIST"))
2180 function list_to_array()
2181 {
2182     #echo "$@"
2183     local COUNT=0
2184     local array
2185     while read -r line; do
2186         array[$COUNT]="$line"
2187         let COUNT++
2188     done <<< $@
2189     echo ${array[@]}
2190 }
2191
2192 -----
2193 -----
2194 -----
2195
2196
2197 File: predictiondecisionengine/trunk/bin/hepspec_crop_results.sh
2198 #!/bin/bash
2199
2200 B_PATH=/home/crivella/Fermilab/Benchmarks/hepspec06
2201 KEYS_PATH=/home/crivella/Fermilab
2202
2203 #Adjust this section with your own format used to name instances and keys#####
2204 VMS_DATA=`aws ec2 describe-instances --filters Name>tag-value,Values=grassano` 
2205 VMS=`echo "$VMS_DATA" | grep -F "grassano" | cut -d "\"" -f 4` 
2206 VMS_KEYS=`echo "$VMS_DATA" | grep -F "_grassano" | cut -d "\"" -f 4` 
2207 VMS_PUB_DNS=`echo "$VMS_DATA" | grep -F "PublicDnsName" | cut -d "\"" -f 4 | uniq` 
2208 #####

```

```

2199
2200 B_PATH=/home/crivella/Fermilab/Benchmarks/hepspec06
2201 KEYS_PATH=/home/crivella/Fermilab
2202
2203 #Adjust this section with your own format used to name instances and keys#####
2204 VMS_DATA=`aws ec2 describe-instances --filters Name>tag-value,Values=grassano` 
2205 VMS=`echo "$VMS_DATA" | grep -F "grassano" | cut -d "\"" -f 4` 
2206 VMS_KEYS=`echo "$VMS_DATA" | grep -F "_grassano" | cut -d "\"" -f 4` 
2207 VMS_PUB_DNS=`echo "$VMS_DATA" | grep -F "PublicDnsName" | cut -d "\"" -f 4 | uniq` 
2208 #####

```

```

2211 : 'MANUAL_CHOICES
2212 NUM_VMS=0
2213 while read -r CMD; do
2214     let NUM_VMS++
2215     echo -e "$NUM_VMS-\t$CMD"
2216 done <<< "$VMS"
2217 echo -e "100-\tALL"
2218
2219 CHOICE=0
2220 echo -e "\nChoose VM to get results from: Or type -1 to exit\nFor multiple selection separate
... them with a space"
2221 read CHOICES
2222
2223 FLAG1=1
2224 while [ $FLAG1 -eq 1 ]; do
2225     FLAG1=0
2226     ALL_SELECT=0
2227     if [ $CHOICES -eq -1 ]; then
2228         echo "Quitting...."
2229         exit
2230     fi
2231     for CHOICE in $CHOICES; do
2232         if [ $CHOICE -gt $NUM_VMS ] || [ $CHOICE -lt 0 ] && [ $CHOICE -ne 100 ]; then
2233             FLAG1=1
2234             echo -e "Choice not available..."
2235         fi
2236         if [ $CHOICE -eq 100 ]; then
2237             ALL_SELECT=1
2238         fi
2239     done
2240     if [ $FLAG1 -eq 1 ]; then
2241         echo -e "\nChoose VM to get results from: Or type -1 to exit"
2242         echo "For multiple selection separate them with a space"
2243         read CHOICES
2244     fi
2245 done
2246
2247 if [ $ALL_SELECT -eq 0 ]; then

```

```

2248     for CHOICE in $CHOICES; do
2249         let CHOICE=CHOICE-1
2250         VM=`echo "$VMS" | tail -n $(expr $NUM_VMS - $CHOICE) | head -n 1`
2251         VM_DNS=`echo "$VMS_PUB_DNS" | tail -n $(expr $NUM_VMS - $CHOICE) | head -n 1`
2252         VM_KEY=`echo "$VMS_KEYS" | tail -n $(expr $NUM_VMS - $CHOICE) | head -n 1`
2253
2254         KEY=`find $KEYS_PATH -name "$VM_KEY.pem"`
2255         if [ "$KEY" = "" ]; then
2256             echo "You do not have the necessary key. Terminating..."
2257             exit
2258         fi
2259
2260         mkdir $B_PATH/amazon/${VM:9:-8}
2261         scp -i $KEY -r root@$VM_DNS:/scratch/install/results/* $B_PATH/amazon/${VM:9:-8}/
2262     done
2263 else
2264     for CHOICE in `seq $NUM_VMS`; do
2265         let CHOICE=CHOICE-1
2266         VM=`echo "$VMS" | tail -n $(expr $NUM_VMS - $CHOICE) | head -n 1`
2267         VM_DNS=`echo "$VMS_PUB_DNS" | tail -n $(expr $NUM_VMS - $CHOICE) | head -n 1`
2268         VM_KEY=`echo "$VMS_KEYS" | tail -n $(expr $NUM_VMS - $CHOICE) | head -n 1`
2269         KEY=`find $KEYS_PATH -name "$VM_KEY.pem"`
2270         if [ "$KEY" = "" ]; then
2271             echo "You do not have the necessary key. Terminating..."
2272             exit
2273         fi
2274
2275         mkdir $B_PATH/amazon/${VM:9:-8}
2276         scp -i $KEY -r root@$VM_DNS:/scratch/install/results/* $B_PATH/amazon/${VM:9:-8}/
2277     done
2278 fi
2279
2280 MANUAL_CHOICES'
2281 #AUTO_CHOICES
2282
2283 echo "AutoChoices"
2284 #Amazon cropping

```

```

2286 CHOICES=""
2287 NUM_VMS=0
2288
2289 while read -r CMD; do
2290     let NUM_VMS++
2291     #echo -e "$NUM_VMS-\t$CMD"
2292     if [[ `echo "$CMD" | grep -F "-hepspec"` != "" ]]; then
2293         CHOICES="$CHOICES $NUM_VMS"
2294     fi
2295 done <<< "$VMS"
2296
2297 for CHOICE in $CHOICES; do
2298     VM=`echo "$VMS" | head -n $CHOICE | tail -n 1`
2299     VM_DNS=`echo "$VMS_PUB_DNS" | head -n $CHOICE | tail -n 1`
2300     VM_KEY=`echo "$VMS_KEYS" | head -n $CHOICE | tail -n 1`
2301
2302     KEY=`find $KEYS_PATH -name "$VM_KEY.pem"`
2303     if [ "$KEY" = "" ]; then
2304         echo "You do not have the necessary key for $VM. Terminating..."
2305         exit
2306     fi
2307
2308     mkdir $B_PATH/amazon/${VM:9:-12}
2309     scp -i $KEY -r root@$VM_DNS:/scratch/install/results/* $B_PATH/amazon/${VM:9:-12}/
2310 done
2311
2312 :
2313 CHOICES="fermicloud148"
2314
2315 for CHOICE in $CHOICES; do
2316     mkdir $B_PATH/bare_metal/$CHOICE
2317     scp -r cms_admin@$CHOICE:/hepspec2006/install/results/* $B_PATH/bare_metal/$CHOICE/
2318 done
2319
2320
2321 RESULTS=`find $B_PATH -name "lock.CPU2006" | sort`
2322 while read -r CMD; do
2323     RES_PATH=`dirname $CMD`
```

```

2324 N_CORE=`find $RES_PATH -name "CPU2006*.log" | wc -l`
2325 for COUNT in `seq $N_CORE`; do
2326     if [ $COUNT -eq 1 ]; then
2327         echo "CORE$COUNT" > $RES_PATH/crop_core.txt
2328     else
2329         echo -e "\nCORE$COUNT" >> $RES_PATH/crop_core.txt
2330     fi
2331     cat $RES_PATH/CFP2006.`printf %03d $COUNT`.ref.txt | grep -F "*" | sort | uniq >>
2332     $RES_PATH/crop_core.txt
2333     cat $RES_PATH/CINT2006.`printf %03d $COUNT`.ref.txt | grep -F "*" | sort | uniq >>
2334     $RES_PATH/crop_core.txt
2335     done
2336 done <<< "$RESULTS"
2337 -----
2338 -----
2339
2340
2341 File: predictiondecisionengine/trunk/bin/make_diff_sum.sh
2342 #!/bin/bash
2343 . /home/crivella/bin/functions.sh
2344
2345 RES_PATH=/home/crivella/Fermilab/Benchmarks/s3_stresstest_u/$1
2346 FILE="diff_sum_`date +%F`.log"
2347 RESULTS=`find $RES_PATH -name "diff.log" | sort -n`
2348 while read -r line; do
2349     echo "$line" >> $RES_PATH/$FILE
2350     LIST=`cat $line | grep "Diff for" | cut -d " " -f 1 | cut -d " " -f 3 | sort -n | uniq`
2351     for CMD in $LIST; do
2352         echo -ne "Sum of $CMD simultaneous transfer throughput: \t">>> $RES_PATH/$FILE
2353         LINES=`cat $line | grep "Diff for $CMD is"`
2354         INTEGER=0
2355         DECIMAL=0
2356         while read -r line2; do
2357             VAR=`echo "$line2" | cut -d " " -f 5 | cut -d " " -f 1`
2358             INT=`echo $VAR | cut -d "." -f 1`
2359             if [[ -z "$INT" ]]; then
```

```

2360          INT=0
2361      fi
2362      INTEGER=`echo $INTEGER+$INT | bc -l`
2363      FLOAT=`echo $VAR | cut -d "." -f 2`
2364      DECIMAL=`echo $DECIMAL+$FLOAT | bc -l`
2365      #echo "$INT.$FLOAT -> $INTEGER.$DECIMAL"
2366      done <<< "$LINES"
2367      #echo ##### while [[ $DECIMAL -ge 1000 ]]; do
2368      while [[ $DECIMAL -ge 1000 ]]; do
2369          let DECIMAL-=1000
2370          let INTEGER+=1
2371      done
2372      echo "$INTEGER.\`printf %03d $DECIMAL` GB/s">> $RES_PATH/$FILE
2373      done
2374      echo -e "\n" >> $RES_PATH/$FILE
2375 done <<< "$RESULTS"
2376 exit
2377 -----
2378 -----
2380 -----
2381
2382
2383 File: predictiondecisionengine/trunk/bin/README_aws_launch_benchmark
2384 README for the aws_launch_benchmark.sh script
2385
2386 #####
2387 1. Index
2388      1- Index
2389      2- Required packages
2390      3- Useful variables
2391      4- Optional parameters
2392      5- Priorities
2393      6- Resume mode
2394      7- Adding new benchmarks to the script
2395
2396 #####

```

```

2396 ##### 2. Required packages
2397 2. Required packages
2398     In order to work, this script will check for the presence of the aws and nmap packages. If
2399     one of them is missing, the script will
2400     terminate, asking the user to install them before proceeding.
2401 #####
2402 3. Useful variables
2403     FILES_PATH    Path for the directory containing the run-benchmark_name.sh files
2404     KEYS_PATH     Path for the directory containing the aws keys (.pem)
2405     TMP_PATH      Path for the directory to store the tmp files
2406
2407     benchmark    Name of the chosen benchmark to execute
2408     vol_size     The size of the extra-volumes to create in order to make the specified
2409     benchmark work
2410     ROLE         Role to attach at launch to the instances
2411     PARAM        Additional parameters to pass to the run-$benchmark.sh script
2412     AMI_ID       ID of the AMI to use for the instances
2413     SG_ID        ID of the Security group to use for the instances
2414     KEY          Name of the KEY (without .pem) to use for the instances
2415
2416     benchmarks   Array containing all the benchmark that the script can run (The indexes need
2417     to be matching with those of the
2418         next arrays)
2419     req_vol_size Array containing the sizes of the extra-volumes for each benchmarks
2420     roles        Array containing the role that the instances will need to execute the
2421     benchmark. If a value is null (ex: ("" ""))
2422         then the default "AllowS3_Download" will be selected (required to download
2423         benchmark files from S3)
2424     parameters   Array containing the default parameters to use with each benchmark. They can
2425     be overwritten at launc using the -p
2426         optional parameter (See more in paragraph 5)
2427
2428 #####
2429 4. Optional parameters
2430     -a Specify the ID of the AMI the script will try to use

```

```

2426     -s Specify the ID of the Security Group the script will try to use
2427     -k Specify the Name of the Key Pair the script will try to use
2428     -i Specify the kind and amount of instances to benchmark
2429         Example: aws_launch_benchmark.sh -i "3:c3.2xlarge 10:t2.micro 1:m3.4xlarge"
2430             This command will use the later specified benchmark on 3 c3.2xlarge instances, 10
2431             t2.micro and 1 m3.4xlarge
2432             -p Set optional parameters to pass to the "run-name_of_benchmark.sh" executed on the VM
2433                 Example: aws_launch_benchmark.sh -p "1 10 100"
2434                     For bandwidth benchmarks, this will make them do 1 simultaneous up/download of
2435                     files, followed by 10 simultaneous and
2436                     than 100 simultaneous
2437             -z Specify a profile to use, so to operate in different zone than the default one
2438                 The name of the profile should be the same as the zone they are meant for
2439             -r Enable the resume mode (See paragraph 6 for more instruction)
2440 #####
2441 5. Priorities
2442     To specify aws related parameters (AMI ID, KEY, SG_ID, Param, ZONE) there are 3 level of
2443     priorities in this script.
2444     The default values will be the ones specified in the script and will be used unless
2445     optional parameters are being used. In this case
2446         they will override the default values.
2447     The highest level of priorities is based on a check with aws. If the values in use are not
2448     found on aws, the script will prompt the
2449         user and ask if he wants to see a list of the available options on aws to choose from, or
2450         exit
2451 #####
2452 6. Resume mode
2453     The resume mode works only if launching one kind of instances (example: only c3.2xlarge or
2454     only m3.medium ...).
2455     Also, the benchmark the user is launching has to be the same that was previously launched
2456     on the instances he is trying to resume.
2457     Any other use could cause bugs to arise
2458     When the resume mode is enabled, the script will look for a tmp file in the folder

```

```

2453 $TMP_PATH/benchmark_name and read it if found or
2454     exit otherwise.
2455     This file is generated automatically by a previous run of the script, but can be manually
2456     edited if needed
2457         The format of the file has to be
2458             NUM_VMS:INSTANCE_TYPE:INSTANCE_ID:PUB_DNS:KEY      (NUM_VMS start from 0)
2459             example:
2460             0:m3.medium:i-d3ea6708:ec2-52-27-10-197.us-west-2.compute.amazonaws.com:mykey
2461                 1:m3.medium:i-d4ea670f:ec2-52-88-72-102.us-west-2.compute.amazonaws.com:mykey
2462             The data present in the file will be stored in the same arrays used by the script to store
2463             the data of newly created instances, and
2464             will act as the starting point of the other parts of the script depending on the selected
2465             choice that the user will be prompted to      take while in resume mode.
2466             This choice can be either to resume the script from "Volume creation" or from "Start
2467             Benchmark":
2468                 Volume creation      After creating the missing instances, the script will create a
2469                 new volume for all instances, included
2470                     the one in the tmp file. This function is meant to be used, if the script
2471                     for some reason got
2472                         interrupted before the extra-volume was created.
2473                 Start Benchmark      After creating the missing instances, the script will create a
2474                     new volume only for the newly started
2475                         instances. This is meant for launching 2 successive benchmarks, where one
2476                         is the extended version of
2477                             the previous.
2478                         For example. After launching a benchmark on 5 c3.2xlarge instances, the
2479                         user wants to launch the same
2480                             benchmark, but on 10 c3.2xlarge instances. The normal way would be to
2481                             reuse the 5 instances
2482                                 previously launched, and this can be accomplished with the resume mode
2483 #####
2484 7. Adding new benchmarks to the script
2485     1- Modify the benchmarks, req_vol_size, roles, parameters array in the script adding the
2486         name of the benchmark in benchmarks,
2487             the size of the extra-volume needed for it to run in req_vol_size, the name of the role
2488             the benchmark will use in roles (use "")

```

```

2477     to set it to the default one), and the default parameter for that benchmark in
2478     parameters.
2479     Using 0 in req_vol_size will prompt a dynamic allocation of space based on the
2480     parameters. For example while executing a
2481     bandwidth download test, the req_vol_size will depend on the size of the files and
2482     their number. The script consider the
2483     default size for the files of 1GB, and will thus use the sum of the parameters+3 as
2484     the vol_size
2485     2- Load all the required files for your benchmark on s3
2486     3- Modify the section of the script that checks if you have an s3 folder in you bucket
2487     for the benchmark in case you don't want
2488         it to have the same name as the benchmark (example: different benchmarks share the
2489     same files)
2490     4- Create a run-benchmark_name.sh script and put it in the directory conciding with the
2491     var FILES_PATH
2492         This script will be executed as root from the /root/ directory and will need to make
2493     the file sys for the extra-disk and mount
2494         it. It will need afterwards to download the required files from s3 and place them
2495     where needed, and afterwards run all the
2496         commands needed to execute the benchmarks.
2497         See the already write run-* .sh files for examples
2498     Note: The first parameters passed from the aws_launch_benchmark.sh script to the
2499     run-* .sh will always be the time right before going
2500     into the Start Benchmark phase and the number identifying the job. The first is used
2501     to syncronize bandwidth benchmarks.
2502     If the run-* .sh has other parameters, the user will need to take this into account.
2503     Using the "shift" command will shift the
2504     parameter indexes from n to n-1, deleting the $1. This can be a useful way to ignore
2505     the the first 2 default parameters,
2506         enabling a easier use of the $@ to invoke the list of user defined parameters
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
```

File: predictiondecisionengine/trunk/bin/s3\_manager.sh

```

2502#!/bin/bash
2503
2504#D_PATH=$HOME/s3_download
2505#if [[ ! -d $D_PATH ]]; then
2506#    echo "The Download directory doesn't exist. Do you wish to create it on the default
2507#path?(y/n)"
2508#else
2509#    CHOICE="bypass"
2510#fi
2511A_PATH=`find $HOME -name "s3_download_path" 2>/dev/null`
2512D_PATH=`dirname $A_PATH`
2513if [[ "$D_PATH" != "" ]]; then
2514    echo "Download path found as $D_PATH"
2515    CHOICE="bypass"
2516else
2517    echo "The Download directory doesn't exist. Do you wish to create it on the default
2518    path?(y/n)"
2519    D_PATH=$HOME/s3_download
2520fi
2521
2522COUNT=0
2523while [[ "$CHOICE" != "y" && "$CHOICE" != "n" && "$CHOICE" != "bypass" ]]; do
2524    read CHOICE
2525    if [[ "$CHOICE" = "y" ]]; then
2526        if mkdir -p $D_PATH; then
2527            :
2528        else
2529            echo "Cannot create the directory. Exiting..."
2530            exit
2531        fi
2532    elif [[ "$CHOICE" = "n" ]]; then
2533        echo "Do you wish to specify a different path (y) or exit (n)?"
2534        read CHOICE2
2535        if [[ "$CHOICE2" = "y" ]]; then
2536            COUNT2=0
2537            CHECK=0
```

```

2538     while [[ $CHECK -eq 0 ]]; do
2539         echo "Insert the path: "
2540         read D_PATH
2541         if mkdir -p $D_PATH; then
2542             CHECK=1
2543         else
2544             let COUNT2++
2545             echo "Can't create the folder(try number $COUNT2). Enter path again"
2546         fi
2547         if [[ $COUNT2 -ge 3 ]]; then
2548             echo "3 invalid choices. Exiting..."
2549         fi
2550         done
2551
2552         else
2553             exit
2554         fi
2555     else
2556         let COUNT++
2557         echo "Choise is not valid. Enter it again"
2558     fi
2559     if [[ $COUNT -ge 3 ]]; then
2560         echo "3 invalid choices. Exiting..."
2561         exit
2562     fi
2563
2564 done
2565 if [[ -d $D_PATH ]]; then
2566     echo "$D_PATH" > $D_PATH/s3_download_path
2567 fi
2568
2569 echo -e "Select the operaion to perform:\n1- Download files\n2- Upload files\n10-exit"
2570 read CHOICE
2571 if [[ $CHOICE -eq 10 ]]; then
2572     exit
2573 fi
2574
2575 USER=grassano

```

```

2576 BUCKETS_LIST=`aws s3api list-buckets | grep Name | cut -d '"' -f 4`
2577 COUNT=0
2578 BUCKETS[0]=`echo "$BUCKETS_LIST" | wc -l`
2579 while read -r CMD; do
2580     let COUNT++
2581     BUCKETS[$COUNT]=$CMD
2582     echo -e "$COUNT\t$CMD"
2583 done <<< "$BUCKETS_LIST"
2584 echo "Choose the bucket to perform the action on:"
2585 CHOICE2=0
2586 COUNT=0
2587 while [[ $CHOICE2 -lt 1 || $CHOICE2 -gt ${BUCKETS[0]} ]]; do
2588     let COUNT++
2589     if [[ $COUNT -ge 4 ]]; then
2590         echo "3 invalid choices. Exiting..."
2591     fi
2592     read CHOICE2
2593     if [[ $CHOICE2 -lt 1 || $CHOICE2 -gt ${BUCKETS[0]} ]]; then
2594         echo "Choice out of range. Enter again:"
2595     fi
2596 done
2597
2598 if [[ $CHOICE -eq 1 ]]; then
2599     OBJ_DATA=`aws s3api list-objects --bucket ${BUCKETS[$CHOICE2]}`
2600     OBJ_LIST=`echo "$OBJ_DATA" | grep Key | cut -d '"' -f 4`
2601     COUNT=0
2602     OBJ[0]=`echo "$OBJ_LIST" | wc -l`
2603     while read -r CMD; do
2604         let COUNT++
2605         OBJ[$COUNT]=$CMD
2606         echo -e "$COUNT\t$CMD"
2607     done <<< "$OBJ_LIST"
2608     echo "Choose the objects you want to download (num1 num2 ...)"
2609     read DOWNLOADS
2610     for CMD in $DOWNLOADS; do
2611         if [[ $CMD -ge 1 && $CMD -le ${OBJ[0]} ]]; then
2612             echo "Downloading ${OBJ[$CMD]}..."
2613             aws s3 cp s3://${BUCKETS[$CHOICE2]}/${OBJ[$CMD]} $D_PATH

```

```
2614     else
2615         echo "$CMD is an invalid selection. Ignoring it..."
2616     fi
2617 done
2618 elif [[ $CHOICE -eq 2 ]]; then
2619     if [[ $# -gt 0 ]]; then
2620         echo "Taking parameter from function input"
2621         UPLOADS=$@
2622     else
2623         echo "Inser files to upload (name1 name2 ...)"
2624         read UPLOADS
2625     fi
2626     for CMD in $UPLOADS; do
2627         if [[ -f $CMD ]]; then
2628             echo "Uploading $CMD"
2629             aws s3 cp $CMD s3://${BUCKETS[$CHOICE2]}/
2630         else
2631             echo "$CMD is not a regular file. Ignoring it..."
2632         fi
2633     done
2634 fi
2635
2636 exit
2637
2638 -----
2639 -----
2640 -----
```

2641  
2642  
2643

```

1 File: monitoringaccountingbilling/trunk/src/gratia/awsvm/aws-gratia-probe-fix
2 #!/usr/bin/env python
3 import gratia.common.Gratis as Gratia
4 import gratia.common.GratisCore as GratiaCore
5 import gratia.common.GratisWrapper as GratiaWrapper
6 from gratia.common.Gratis import DebugPrint
7 import boto3;
8 from pprint import pprint;
9 import datetime
10 from time import mktime
11 import time
12 import sys
13 from gratia.awsrm.spot_price import spot_price
14 from gratia.awsrm.cpuutil import cpuUtilization
15 from gratia.awsrm.inst_hardware import insthardware
16 from gratia.awsrm.get_account_id import get_account_id
17 import os
18
19 class Awsgratiaprobe:
20     def __init__(self):
21         GratiaCore.Config.set_DebugLevel(5)
22
23
24     def process(self):
25         ec2=boto3.client('ec2',region_name='us-west-2')
26             mygacid=get_account_id()
27             owneracct=mygacid.get_id()
28         response = ec2.describe_instances()
29         #pprint(response)
30         resv=response['Reservations']
31         for reservation in resv:
32             #pprint(reservation)
33             instances=reservation['Instances']
34             for instance in instances:
35                 #pprint(instance)
36                 print instance['InstanceId']
37                 #print instance['InstanceType']
38                 r = Gratia.UsageRecord()

```

```

39             user="aws account user"
40             project="aws-no project name given"
41             voname="fermilab"
42             try:
43                 tags=instance['Tags']
44                 print "the tags are"
45                 for tag in tags:
46                     print tag['Key'],
47                     print tag['Value']
48                 for tag in tags:
49                     if tag['Key'].lower() == 'user'.lower():
50                         user=tag['Value']
51                 for tag in tags:
52                     if tag['Key'].lower() == 'name'.lower():
53                         r.JobName(tag['Value'])
54                 for tag in tags:
55                     if tag['Key'].lower() == 'project'.lower():
56                         project=tag['Value']
57 #
58                 voname=tag['Value']
59                 for tag in tags:
60                     if tag['Key'].lower() == 'vo'.lower() or tag['Key'].lower() ==
61 "...voname".lower():
62                         voname=tag['Value']
63             except KeyError:
64                 print 'no tags'
65             r.LocalUserId(user)
66             r.GlobalUserId(owneracct)
67             r.ProjectName(project)
68             r.VOName(voname)
69             r.ReportableVOName(voname)
70             #Public Ip address is retrieved if instance is running"
71             try:
72                 ipaddr=instance['PublicIpAddress']
73                 r.MachineName(instance['PublicIpAddress'],instance['ImageId'])
74             except KeyError:
75                 r.MachineName("no Public ip as instance has been stopped",instance['ImageId'])

```

```

76             r.LocalJobId(instance['InstanceId'])
77             r.GlobalJobId(instance['InstanceId']+ "#" +repr(time.time()))
78     #print 'hello1'
79     try:
80         for tag in tags:
81             if tag['Key'].lower() == 'name'.lower():
82                 r.JobName(tag['Value'])
83     except KeyError:
84             print 'no tags'
85     #status,description=recs[i].getStatus()
86
87     state=instance['State']
88     #print state['Name']
89     status=1
90     if state['Name']=="running":
91         print status
92         status=0
93     else:
94         status=1
95     #print 'hello'
96     #print status
97     pprint(r)
98     #print instance['StateTransitionReason']
99     description=instance['StateTransitionReason']
100    #print description
101    r.Status(status,description)
102
103 #r.ProcessorsDescription(instance['InstanceType'])
104 #r.MachineNameDescription(instance['ImageId'])
105
106     try:
107         ipaddr=instance['PrivateIpAddress']
108
109     r.Host(instance['PrivateIpAddress'],False,instance['Placement']['AvailabilityZone'])
110
111     r.SubmitHost(instance['PrivateIpAddress'],instance['Placement']['AvailabilityZone'])
112     except KeyError:
113         r.SubmitHost("no Private ip as instance has been terminated")

```

```

112     #print 'setting site name'
113
114     #GratiaCore.Config.setSiteName('aws'+instance['Placement']['AvailabilityZone'])
115     #print 'done setting'
116     #r.ReportedSiteName('aws'+instance['Placement']['AvailabilityZone'])
117     r.ResourceType('aws')
118     r.Njobs(1,"The no of jobs running at a time")
119     r.NodeCount(1) # default to total
120     try:
121         hardwdet=GratiaCore.Config.getConfigAttribute("HardwareDetails")
122     except Exception as e:
123         print "The File for hardware details is not present. Pls check config file
124 Hardware details value."
125     else:
126         hardwdet="/usr/share/gratia/awsvm/hardwareinst"
127         instdet=insthardware(hardwdet)
128         types=instdet.gettypedetails()
129         pprint(types)
130         processor='1'
131         memory=''
132         price=0.0
133         for t in types:
134             print t['instance-type'], instance['InstanceType']
135             if t['instance-type'] == instance['InstanceType']:
136                 pprint (t)
137                 processor=t['vcpu']
138                 memory=t['ram']
139                 price=t['price']
140                 print processor,memory,price
141                 #print memory," the value of memory"
142                 cpu=float(processor)
143                 print cpu
144                 r.Processors(int(cpu),0,"total",instance['InstanceType'])
145                 r.Memory(float(memory))
146                 chargedesc=""
147                 # Spot price is retrieved using instance id as the charge per hour of that
148                 # instance in the last hour
149                 #print instance['InstanceId']

```

```

148         if "SpotInstanceRequestId" in instance.keys():
149             sp=spot_price.spot_price()
150             value=sp.get_price(instance['InstanceId'])
151             #print value
152             price=value
153             chargedesc="The instance is a on-demand instance hence charge is fixed per
... hour"
154             if status==1:
155                 price=0
156             chargedesc="The spot price charged in last hour corresponding to launch time"
157             r.Charge(str(price),"$","$/instance hr",chargedesc)
158             # The Time period for which the spot price and other values are calculated is noted
... down
159             launchtime=instance['LaunchTime']
160             #print launchtime.hour
161             #print 'hello3'
162             minu=launchtime.minute
163             #print minu
164
165             currtime=time.time()
166
167
168             EndTime=datetime.datetime.now()
169             #print type(EndTime)
170             EndTime=EndTime.replace(minute=minu)
171             StartTime=EndTime.replace(hour=(EndTime.hour-1))
172             #print StartTime
173             #print EndTime
174             #print 'starttime'
175             t=StartTime.date()
176             #print GratiaCore.TimeToString(time.mktime(t.timetuple()))
177             #print 'convert'
178             stime=time.mktime(StartTime.timetuple())
179             r.StartTime(stime)
180
181             et=EndTime.date()
182             etime=time.mktime(EndTime.timetuple())
183             r.EndTime(etime)

```

```

184
185             #print 'hello123'
186             #print etime-stime," the diff"
187             r.WallDuration(etime-stime)
188             Cpu=cpuUtilization()
189             aver=Cpu.getUtilPercent(instance['InstanceId'])
190             #print aver," average in percentage"
191             #print type(aver)
192             #print type(cpu)
193             if aver is None:
194                 cpuUtil=0.0
195                 print "The CPU Utilization value is NULL as the instance was not running in the
... last hour"
196                 r.CpuDuration(0,'user')
197             else:
198                 cpuUtil=aver
199                 r.CpuDuration((etime-stime)*float(aver)*cpu/100,'user')
200                 r.CpuDuration(0,'system')
201                 r.ResourceType("AWSVM")
202                 r.AdditionalInfo("Version","1.0")
203
204             #print r
205             print 'done'
206             Gratia.Send(r)
207
208
209
210 if __name__ == '__main__':
211     try:
212         Gratia.Initialize('/etc/gratia/awsvm/ProbeConfig')
213         GratiaWrapper.CheckPreconditions()
214         vmProbe=Awsgratiaprobe()
215         Filelock="filelock"
216         conf=GratiaCore.Config
217         duplicateLock=conf.getConfigAttribute("ExemptDuplicates")
218         filelock=conf.getConfigAttribute("DuplicateFilelock")
219         if duplicateLock == "True":
220             if os.path.isfile(Filelock):

```

```

221         fl=open(Filelock, 'r+')
222         date=datetime.datetime.now()
223         line=fl.readline()
224         print line
225         prevdate = datetime.datetime.strptime(line, "%Y-%m-%d %H:%M:%S.%f")
226         print prevdate
227         currtime=time.mktime(date.timetuple())
228         prevtime=time.mktime(prevdate.timetuple())
229         diff=currtime-prevtime
230         print diff
231         if diff>=3599.0:
232             fl.seek(0, 0)
233             fl.truncate()
234             fl.write(str(date));
235             t = os.path.getmtime(Filelock)
236             print t
237             print datetime.datetime.fromtimestamp(t)
238             fl.close()
239             vmProbe.process()
240         else:
241             print "hour is not over yet"
242             fl.close()
243     else:
244         fl=open(Filelock,'w+')
245         date=str(datetime.datetime.now())
246         fl.write(date);
247         fl.close()
248         vmProbe.process()
249     else:
250         vmProbe.process()
251 except Exception, e:
252     print >> sys.stderr, str(e)
253     sys.exit(1)
254 sys.exit(0)
255
256
257 -----
258 -----

```

```

259 -----
260
261
262 File: monitoringaccountingbilling/trunk/src/gratia/awsvm/aws-gratia-probe-multi-fix
263 #!/usr/bin/env python
264 import gratia.common.Gratis as Gratia
265 import gratia.common.GratisCore as GratiaCore
266 import gratia.common.GratisWrapper as GratiaWrapper
267 from gratia.common.Gratis import DebugPrint
268 import boto3;
269 from boto3.session import Session
270 from pprint import pprint;
271 import datetime
272 from time import mktime
273 import time
274 import sys
275 from gratia.awsvm.spot_price import spot_price
276 from gratia.awsvm.cpuutil import cpuUtilization
277 from gratia.awsvm.inst_hardware import insthardware
278 from gratia.awsvm.get_account_id import get_account_id
279 import os
280
281 class Awsgratiaprobe:
282     def __init__(self):
283         GratiaCore.Config.set_DebugLevel(5)
284
285
286     def process(self):
287         for account in ( 'rnd', 'cms', 'nova', 'fermilab' ):
288             session = Session(profile_name = account)
289             for region in ( 'us-west-2','us-west-1','us-east-1'):
290                 ec2=boto3.client('ec2',region_name=region)
291                 mygacid=get_account_id()
292                 owneracct=mygacid.get_id()
293                 print owneracct
294                 response = ec2.describe_instances()
295                 #pprint(response)
296                 resv=response['Reservations']

```

```

297     for reservation in resv:
298         #pprint(reservation)
299         instances=reservation['Instances']
300         for instance in instances:
301             #pprint(instance)
302             print instance['InstanceId']
303             #print instance['InstanceType']
304             r = Gratia.UsageRecord()
305                 # set the defaults
306             user="aws account user"
307             project="aws-no project name given"
308             vname="fermilab"
309             try:
310                 tags=instance['Tags']
311                 print "the tags are"
312                 for tag in tags:
313                     print tag['Key'],
314                     print tag['Value']
315                 for tag in tags:
316                     if tag['Key'].lower() == 'user'.lower():
317                         user=tag['Value']
318                     for tag in tags:
319                         if tag['Key'].lower() == 'name'.lower():
320                             r.JobName(tag['Value'])
321                     for tag in tags:
322                         if tag['Key'].lower() == 'project'.lower():
323                             project=tag['Value']
324                     for tag in tags:
325                         if tag['Key'].lower() == 'vo'.lower() or tag['Key'].lower() ==
326                         ... 'vname'.lower():
327                             vname=tag['Value']
328             except KeyError:
329                 print 'no tags'
330             r.LocalUserId(user)
331             r.GlobalUsername(owneracct)
332             r.ProjectName(project)
333             r.VOName(vname)
334             r.ReportableVOName(vname)

```

```

334             "#Public Ip address is retrieved if instance is running"
335             try:
336                 ipaddr=instance['PublicIpAddress']
337                 r.MachineName(instance['PublicIpAddress'],instance['ImageId'])
338             except KeyError:
339                 r.MachineName("no Public ip as instance has been
340     ... stopped",instance['ImageId'])
341
342             r.LocalJobId(instance['InstanceId'])
343             r.GlobalJobId(instance['InstanceId']+="#" +repr(time.time()))
344             #print 'hello1'
345             try:
346                 for tag in tags:
347                     if tag['Key'].lower() == 'name'.lower():
348                         r.JobName(tag['Value'])
349             except KeyError:
350                 print 'no tags'
351             #status,description=recs[i].getStatus()
352
353             state=instance['State']
354             #print state['Name']
355             #set the default status
356             status=1
357             if state['Name']=="running":
358                 print status
359                 status=0
360             else:
361                 status=1
362             #print 'hello'
363             #print status
364             pprint(r)
365             #print instance['StateTransitionReason']
366             description=instance['StateTransitionReason']
367                 #print description
368             r.Status(status,description)
369
370             #r.ProcessorsDescription(instance['InstanceType'])

```

```

371         #r.MachineNameDescription(instance['ImageId'])
372
373     try:
374         ipaddr=instance['PrivateIpAddress']
375
376     r.Host(instance['PrivateIpAddress'],False,instance['Placement']['AvailabilityZone'])
377
378     r.SubmitHost(instance['PrivateIpAddress'],instance['Placement']['AvailabilityZone'])
379     except KeyError:
380         r.SubmitHost("no Private ip as instance has been terminated")
381     #print 'setting site name'
382
383     #GratiaCore.Config.setSiteName('aws'+instance['Placement']['AvailabilityZone'])
384     #print 'done setting'
385     #r.ReportedSiteName('aws'+instance['Placement']['AvailabilityZone'])
386     r.ResourceType('aws')
387     r.Njobs(1,"The no of jobs running at a time")
388     r.NodeCount(1) # default to total
389     try:
390         hardwdet=GratiaCore.Config.getConfigAttribute("HardwareDetails")
391     except Exception as e:
392         print "The File for hardware details is not present. Pls check config
393     file Hardware details value."
394     else:
395         hardwdet="/usr/share/gratia/awsvm/hardwareinst"
396         instdet=insthardware(hardwdet)
397         types=instdet.gettypedetails()
398         pprint(types)
399         processor='1'
400         memory=''
401         price=0.0
402         for t in types:
403             print t['instance-type'], instance['InstanceType']
404             if t['instance-type'] == instance['InstanceType']:
405                 pprint (t)
406                 processor=t['vcpu']
407                 memory=t['ram']

```

```

408
409         price=t['price']
410         #print processor,memory,price
411         #print memory," the value of memory"
412         cpu=float(processor)
413         #print cpu
414         r.Processors(int(cpu),0,"total",instance['InstanceType'])
415         r.Memory(float(memory))
416         chargedesc=""
417         # Spot price is retrieved using instance id as the charge
418         # per hour of that instance in the last hour
419         #print instance['InstanceId']
420         if "SpotInstanceRequestId" in instance.keys():
421             sp=spot_price.spot_price()
422             value=sp.get_price(instance['InstanceId'])
423             #print value
424             price=value
425             chargedesc="The instance is a on-demand instance hence charge is fixed
426             per hour"
427             if status==1:
428                 price=0
429                 chargedesc="The spot price charged in last hour corresponding to launch
430             time"
431             r.Charge(str(price),"$","$/instance hr",chargedesc)
432             # The Time period for which the spot price and other values are calculated
433             is noted down
434             launchtime=instance['LaunchTime']
435             #print launchtime.hour
436             minu=launchtime.minute
437             #print minu
438             currtime=time.time()

        EndTime=datetime.datetime.now()
        EndTime=EndTime.replace(minute=minu)
        StartTime=EndTime.replace(hour=(EndTime.hour-1))
        #print 'starttime'
        t=StartTime.date()

```

```

439         #print GratiaCore.TimeToString(time.mktime(t.timetuple()))
440         #print 'convert'
441         stime=time.mktime(StartTime.timetuple())
442             r.StartTime(stime)
443
444             et=EndTime.date()
445             etime=time.mktime(EndTime.timetuple())
446             r.EndTime(etime)
447                 r.WallDuration(etime-stime)
448             Cpu=cpuUtilization()
449             aver=Cpu.getUtilPercent(instance['InstanceId'])
450             #print aver," average in percentage"
451             #print type(aver)
452             #print type(cpu)
453             if aver is None:
454                 cpuUtil=0.0
455                 print "The CPU Utilization value is NULL as the instance was not
... running in the last hour"
456                     r.CpuDuration(0,'user')
457             else:
458                 cpuUtil=aver
459                 r.CpuDuration((etime-stime)*float(aver)*cpu/100,'user')
460                 r.CpuDuration(0,'system')
461                     r.ResourceType("AWSVM")
462                     r.AdditionalInfo("Version","1.0")
463
464             #print r
465             print 'done'
466             Gratia.Send(r)
467
468
469
470
471 if __name__ == '__main__':
472     try:
473         Gratia.Initialize('/etc/gratia/awsvm/ProbeConfig')
474         GratiaWrapper.CheckPreconditions()
475         vmProbe=Awsgratiaprobe()

```

```

476     Filelock="filelock"
477     conf=GratiaCore.Config
478     duplicatelock=conf.getConfigAttribute("ExemptDuplicates")
479     filelock=conf.getConfigAttribute("DuplicateFilelock")
480     if duplicatelock == "True":
481         if os.path.isfile(Filelock):
482             fl=open(Filelock, 'r+')
483             date=datetime.datetime.now()
484             line=fl.readline()
485             print line
486             prevdate = datetime.datetime.strptime(line, "%Y-%m-%d %H:%M:%S.%f")
487             print prevdate
488             currtime=time.mktime(date.timetuple())
489             prevtime=time.mktime(prevdate.timetuple())
490             diff=currtime-prevtime
491             print diff
492             if diff>=3599.0:
493                 fl.seek(0, 0)
494                 fl.truncate()
495                 fl.write(str(date));
496                 t = os.path.getmtime(Filelock)
497                 print t
498                 print datetime.datetime.fromtimestamp(t)
499                 fl.close()
500                 vmProbe.process()
501             else:
502                 print "hour is not over yet"
503                 fl.close()
504             else:
505                 fl=open(Filelock, 'w+')
506                 date=str(datetime.datetime.now())
507                 fl.write(date);
508                 fl.close()
509                 vmProbe.process()
510             else:
511                 vmProbe.process()
512             except Exception, e:
513                 print >> sys.stderr, str(e)

```

```

514         sys.exit(1)
515     sys.exit(0)
516
517
518 -----
519 -----
520 -----
521
522
523 File: monitoringaccountingbilling/trunk/src/gratia/awsvm/awsvm_probe.cron.sh
524 python /usr/share/gratia/awsvm/aws-gratia-probe
525
526 -----
527 -----
528 -----
529
530
531 File: monitoringaccountingbilling/trunk/src/gratia/awsvm/hardwareinst
532 instance-type vcpu ram price
533 m3.medium 1 3.75 0.067
534 m3.large 2 7.5 0.133
535 t2.micro 1 1 0.013
536 m3.xlarge 4 15 0.266
537 t2.small 1 2 0.026
538 t2.medium 2 4 0.052
539 t2.large 2 8 0.104
540 m4.large 2 8 0.126
541 m4.2xlarge 8 32 0.504
542 m4.xlarge 4 16 0.252
543 m4.4xlarge 16 64 1.008
544 m4.10xlarge 40 160 2.52
545 m3.2xlarge 8 30 0.532
546 c4.large 2 3.75 0.11
547 c4.xlarge 4 7.5 0.22
548 c4.2xlarge 8 15 0.441
549 c4.4xlarge 16 30 0.882
550 c4.8xlarge 36 60 1.763
551 c3.large 2 3.75 0.105

```

```

552 c3.xlarge 4 7.5 0.21
553 c3.2xlarge 8 15 0.42
554 c3.4xlarge 16 30 0.84
555 c3.8xlarge 32 60 1.68
556 g2.2xlarge 8 15 0.65
557 g2.8xlarge 32 60 2.6
558 r3.large 2 15 0.175
559 r3.xlarge 4 30.5 0.35
560 r3.2xlarge 8 61 0.7
561 r3.4xlarge 16 122 1.4
562 r3.8xlarge 32 244 2.8
563 i1.xlarge 4 30.5 0.853
564 i2.2xlarge 8 61 1.705
565 i2.4xlarge 16 122 3.41
566 i2.8xlarge 32 244 6.82
567 d2.xlarge 4 30.5 0.69
568 d2.2xlarge 8 61 1.38
569 d2.4xlarge 16 56 2.76
570 d2.8xlarge 36 244 5.52
571
572 -----
573 -----
574 -----
575
576
577 File: monitoringaccountingbilling/trunk/src/gratia/awsvm/README
578 Steps to install AWS-gratia-probe(procedures for SL6)(not supported for SL5):
579 You could also follow
... https://twiki.grid.iu.edu/bin/view/Documentation/Release3/InstallAwsVmGratiaProbe
580
581 1. First install the EPEL 6
582     rpm -Uvh http://dl.fedoraproject.org/pub/epel/5/i386/epel-release-5-4.noarch.rpm
583 2. Next you need to get the osg repo
584     http://repo.grid.iu.edu/osg/3.2/osg-3.2-el6-release-latest.rpm
585     and use yum to install the gratia-probe-awsvm.
586     The rpm name being gratia-probe-awsvm-(% VERSION).el6.noarch.rpm
587 3. You have to install pip and boto3
588 B) yum install -y python-pip

```

```

589 C) Next install the boto3 package using pip
590     pip install boto3
591 4. Boto3 needs the aws credentials file containing the IAM Access ID and Key inorder to access the
... AWS console.
592 A) create a credentials file in the standard place specified by AWS
593 5. Next modify the ProbeConfig file
594     vi /etc/gratia/awsvm/ProbeConfig
595 A) Modify the host name and ports to point to the gratia server like below
596     CollectorHost="fermicloud054.fnal.gov:8880"
597         SSLHost="fermicloud054.fnal.gov:8443"
598     SSLRegistrationHost="fermicloud054.fnal.gov:8080"
599 B) Also enable the probe by setting below to one
600     EnableProbe="1"
601 c) Also enable Exempt duplicates in order to prevent probe from sending records more than once
... per hour using
602     ExemptDuplicates="1"
603 Save the changes to config file
604 6. Next we need to add the probe's cron to the system cron and enable it
605 A) add the cron
606     chkconfig --add gratia-probes-cron
607 B) Enable the cron
608     chkconfig gratia-probes-cron on
609 c) Check its status
610     chkconfig --list gratia-probes-cron
611 D) also start the service
612     service gratia-probes-cron start
613 7. The aws-gratia-probe is started and will run every one hour
614 we can check the logs in
615     vi /var/logs/gratia/(%date).log
616
617
618
619 ALSO INCASE ANY ERRORS ARE THROWN BY THE PROBE DURING RUNTIME
620 OR IF THE PROBE DOESNT EXECUTE PROPERLY
621 1. IF the exempt duplicates is set and the filelock throws errors
622     check whether the filelock file exists at
623     /var/lib/gratia/filelock
624     and also where it is set properly in ProbeConfig file

```

```

625     DuplicateFilelock="/var/lib/gratia/filelock"
626 2. If it says hardwareinst doesnt exist check whether the file exists and the value is it properly
... in probe
627     HardwareDetails="/usr/share/gratia/awsvm/hardwareinst"
628 3. If u need to add or update the values to the table in hardwareinst plf follow the syntax
629     instance-type(\tab) vcpu(\tab) ram(\tab) price(\tab)(\newline)
630     value1(\tab) value2(\tab) value3(\tab) value4(\tab)(\newline)
631     eg : m3.medium(\tab) 1(\tab) 3.75(\tab) 0.067(\tab)(\newline)
632
633
634 -----
635 -----
636 -----
637
638
639 File: monitoringaccountingbilling/trunk/src/gratia/site-packages/gratia/awsvm/__init__.py
640
641 -----
642 -----
643 -----
644
645
646 File: monitoringaccountingbilling/trunk/src/gratia/site-packages/gratia/awsvm/cpuutil.py
647 #!/usr/bin/env python
648 from pprint import pprint
649 import datetime;
650 import boto3;
651 import sys
652 import os
653 import time
654
655 class cpuUtilization:
656     def __init__(self):
657         self.ec2=boto3.client('ec2',region_name='us-west-2')
658     def getUtilPercent(self,instdid):
659         ec2=boto3.client('ec2',region_name='us-west-2')
660         cw = boto3.client('cloudwatch',region_name='us-west-2')
661         resp=ec2.describe_instances(InstanceIds=[instdid])

```

```

662     #pprint(resp)
663     #print 'hello'
664     resv=resp['Reservations']
665     for reservation in resv:
666         #pprint(reservation)
667         instances=reservation['Instances']
668         for instance in instances:
669             #pprint(instance)
670             #print instance['LaunchTime']
671             launchtime=instance['LaunchTime']
672             zone=instance['Placement']['AvailabilityZone']
673             #print zone
674             inst_type=instance['InstanceType']
675             print inst_type
676             #print launchtime.hour
677             minu=launchtime.minute
678             #print minu
679             EndTime=datetime.datetime.utcnow()
680             EndTime=EndTime.replace(minute=minu)
681             StartTime=EndTime.replace(hour=(EndTime.hour-1))
682             print StartTime
683             print EndTime
684
685             response = cw.get_metric_statistics(
686                 Namespace='AWS/EC2',
687                 MetricName='CPUUtilization',
688                 Dimensions=[
689                     {
690                         'Name': 'InstanceId',
691                         'Value': instid
692                     },
693                 ],
694                 StartTime=StartTime,
695                 EndTime=EndTime,
696                 Period=3600,
697                 Statistics=['Average','Minimum','Maximum'],
698                 Unit='Percent')
699             #pprint(response)

```

```

700         datapoints=response['Datapoints']
701         if len(datapoints)==1:
702             datapoint=datapoints[0]
703             average=datapoint['Average']
704             #print average
705             return average
706
707
708 -----
709 -----
710 -----
711
712
713 File: monitoringaccountingbilling/trunk/src/gratia/site-packages/gratia/awsvm/get_account_id.py
714 #!/usr/bin/env python
715 from pprint import pprint;
716 import datetime;
717 import boto3;
718 import sys
719 import os
720 import time
721
722 class get_account_id:
723     def __init__(self):
724         self.ec2=boto3.client('ec2',region_name='us-west-2')
725
726     def get_id(self):
727         myec2=boto3.client('ec2',region_name='us-west-2')
728         resp=myec2.describe_images(Owners=['self'])
729         #pprint(resp)
730         images=resp['Images']
731         for image in images:
732             myowner=image['OwnerId']
733             #as written this will return the last OwnerID
734             #but they should all be the same
735         return myowner
736
737 -----

```

```

738 -----
739 -----
740
741
742 File: monitoringaccountingbilling/trunk/src/gratia/site-packages/gratia/awsVm/inst_hardware.py
743 #!/usr/bin/env python
744 from pprint import pprint
745 import datetime;
746
747 class insthardware:
748     def __init__(self,filelocation):
749         self.fileloc=filelocation
750     def gettypedetails(self):
751         types=[]
752         field=[]
753         #repo = {}
754
755         infile = open(self.fileloc,'r')
756         firstline = infile.readline()
757         fields=firstline.split("\t")
758         #print fields
759         for f in fields:
760             #print f
761             if f == "\n":
762                 fields.remove(f)
763
764             #print fields
765         lines= infile.readlines()
766         #print firstline
767         for i in lines:
768             #print i
769             values=i.split("\t")
770             values.remove("\n")
771             #print values
772             x=0
773             repo={}
774             while x<len(fields):
775                 repo[fields[x]]=values[x]

```

```

776             x+=1
777             #print repo
778             types.append(repo)
779             #pprint(types)
780             return types
781
782             #module = '.join(i.split(',')[:-1])
783             #time = '.join(i.split(',')[1:]).replace('\n','')
784             #if not module in repo:
785             #    repo[module] = time
786
787 -----
788 -----
789 -----
790
791
792 File: monitoringaccountingbilling/trunk/src/gratia/site-packages/gratia/awsVm/spot_price.py
793 #!/usr/bin/env python
794 import datetime;
795 import boto3;
796 import sys
797 import os
798 import time
799
800 class spot_price:
801     def __init__(self):
802         self.ec2=boto3.client('ec2',region_name='us-west-2')
803
804     def get_price(self,instdid):
805         resp=self.ec2.describe_instances(InstanceIds=[instdid])
806         #pprint(resp)
807         #print 'hello'
808         resv=resp['Reservations']
809         for reservation in resv:
810             #pprint(reservation)
811             instances=reservation['Instances']
812             for instance in instances:
813                 #pprint(instance)

```

```

814     #print instance['LaunchTime']
815     launchtime=instance['LaunchTime']
816     zone=instance['Placement']['AvailabilityZone']
817     #print zone
818     inst_type=instance['InstanceType']
819     #print inst_type
820     #print launchtime.hour
821     minu=launchtime.minute
822     #print minu
823     StartTime=datetime.datetime.utcnow()
824     #print StartTime
825     if(minu>StartTime.minute):
826         StartTime=StartTime.replace(hour=(StartTime.hour-1))
827     StartTime=StartTime.replace(minute=minu)
828     #print StartTime
829     EndTime=StartTime
830     #response =
831     - self.ec2.describe_spot_price_history(StartTime=datetime(2015,7,8,9,00,00),EndTime=datetime(2015,7,8
832     - ,9,00,00),InstanceTypes=['m3.medium'],ProductDescription=['Linux/UNIX'],AvailabilityZone='us-west-
833     - 2a',NextToken='abc')
834     #pprint(response)
835     #print 'hello'
836     resp1 = self.ec2.describe_spot_price_history(
837         DryRun=False,
838         StartTime=StartTime,
839         EndTime=EndTime,
840         InstanceTypes=[inst_type],
841         ProductDescriptions=['Linux/UNIX'],
842         Filters=[],
843         AvailabilityZone=zone,
844         MaxResults=1000,
845         NextToken='')
846     #pprint(resp1)
847     sphs=resp1['SpotPriceHistory']
848     if len(sphs)==0:
849         print "no spot price history as Instance is not of a spot Instance type"
850         spotprice=0
851     else:

```

```

849         sph=sphs[0]
850         spotprice=sph['SpotPrice']
851         #print spotprice
852         #print 'hello'
853         return spotprice
854         return StartTime
855 if __name__ == '__main__':
856     try:
857         sp=spot_price()
858         value=sp.get_price('i-d0952f26')
859         #print value
860     except IndexError:
861         print "The instance is not a spot type instance and Hence there is no spot price history"
862     except Exception, e:
863         print >> sys.stderr, str(e)
864         sys.exit(1)
865     sys.exit(0)
866
867 -----
868 -----
869 -----
870
871
872 File: monitoringaccountingbilling/trunk/src/monitor/aws.py
873 #!/usr/bin/python
874 from collections import defaultdict
875 from optparse import OptionParser
876 import time
877 import logging
878 import datetime
879
880 import boto3
881
882 from graphite import Graphite
883
884 def get_ec2_instance_cpu(region,instance,end_time=None,period=300):
885     if end_time is None:
886         end_time=datetime.datetime.utcnow()

```

```

887     cw = boto3.client('cloudwatch',region_name=region)
888     response = cw.get_metric_statistics(
889         Namespace='AWS/EC2',
890         MetricName='CPUUtilization',
891         Dimensions=[{'Name':'InstanceId','Value':instance}],
892         StartTime=end_time-datetime.timedelta(seconds=period*2),
893         EndTime=end_time,
894         Period=period,
895         Statistics=['Average','Minimum','Maximum'],
896         Unit='Percent')
897     datapoints=response['Datapoints']
898     r = {}
899     if len(datapoints) > 0:
900         datapoint=datapoints[-1]
901         r['avg'] = datapoint['Average']
902         r['min'] = datapoint['Minimum']
903         r['max'] = datapoint['Maximum']
904     else:
905         logging.warning('no CPU utilization received for instance %s'%instance)
906     return r
907
908 def get_ec2_instances(region):
909     r = defaultdict(int)
910     cpu = defaultdict(float)
911     #try:
912     ec2 = boto3.resource('ec2',region)
913     instances = ec2.instances.all()
914     for i in instances:
915         type = i.instance_type.replace('.','_')
916         if i.state['Name'] == 'running':
917             cpu_usage = get_ec2_instance_cpu(region, i.instance_id)
918             cpu[type] += cpu_usage.get('avg',0)
919             metric = "{0}.counts.{1}".format(type,i.state['Name'])
920             r[metric] += 1
921
922     for k,v in cpu.iteritems():
923         r[k+".cpu.avg"] = v/r[k+".counts.running"]
924     #except Exception as e:

```

```

925     #     logging.error('error communicating with AWS: %s'%e)
926     #     raise e
927     return r
928
929 if __name__ == '__main__':
930     parser = OptionParser()
931     parser.add_option('-t','--test',action="store_true",
932                     dest="test",default=False,
933                     help="output data to stdout, don't send to graphite. Implies -1.")
934     parser.add_option('-1','--once',action="store_true",
935                     dest="once",default=False,
936                     help="run once and exit")
937     parser.add_option('-r','--region', default="us-west-2",
938                     help="AWS region to query; default=us-west-2")
939     parser.add_option('-n','--namespace', default="test",
940                     help="base graphite metric namespace; default=test")
941     parser.add_option('-m','--meta_namespace', default="probes.aws_instances",
942                     help="probe metadata graphite metric namespace; default=probes.aws_instances")
943     parser.add_option('-i','--interval', type="int", default=240,
944                     help="post interval (seconds); default=240")
945
946     (opts,args) = parser.parse_args()
947
948     logformat="%(asctime)s - %(name)s - %(levelname)s - %(message)s"
949     if opts.test:
950         logging.basicConfig(format=logformat,level=logging.DEBUG)
951     else:
952         logging.basicConfig(format=logformat,level=logging.INFO)
953
954     g = Graphite()
955     while True:
956         start = time.time()
957         data = get_ec2_instances(opts.region)
958         duration = time.time()-start
959         logging.info("queried AWS region {0} in {1} s".format(opts.region,duration))
960
961         send_start = time.time()
962         g.send_dict(opts.namespace, data, debug_print=opts.test, send_data=(not opts.test))

```

```
963     meta_data = {
964         "update_time": duration,
965         }
966     g.send_dict(opts.meta_namespace, meta_data, debug_print=opts.test, send_data=(not
... opts.test))
967
968     duration = time.time()-send_start
969     logging.info("sent {0} metrics to graphite namespace {1} in {2}"
... s".format(len(data)+len(meta_data), opts.namespace, duration))
970
971     if opts.test or opts.once:
972         break
973
974     duration = time.time()-start
975     sleep = max(opts.interval-duration-10,0)
976     logging.info("sleeping {0} s".format(sleep))
977     time.sleep(sleep)
978
979 -----
980 -----
981 -----
```

```

1 File: ondemandservices/hepcloud-init-workernode
2 #!/bin/sh
3 ### BEGIN INIT INFO
4 # chkconfig: 2345 27 25
5 # Provides:          hepcloud-init-workernode
6 # Required-Start:    $local_fs $network
7 # Should-Start:     $time
8 # Required-Stop:
9 # Should-Stop:
10 # Default-Start:   2 3 4 5
11 # Default-Stop:    0 1 6
12 # Short-Description: Fix host name and az-specific config files
13 # Description:      Start cloud-init and runs the initialization phase
14 #                   and any associated initial modules as desired.
15 #test
16 ### END INIT INFO
17 LOG="/var/log/hepcloud-init-workernode.log"
18 RETVAL=0
19
20 prog="hepcloud-init-workernode"
21
22 start() {
23
24 touch /var/lock/subsys/hepcloud-init-workernode
25 #
26 # get the public hostname of the EC2 instance and change the
27 # output of the hostname command to match that. (needed for gridftp).
28 #
29 mypublicip=`GET http://169.254.169.254/latest/meta-data/public-ipv4`
30 myrc=$?
31 if [ $myrc -ne 0 ]
32 then
33     echo "My public IP not defined" >> $LOG
34     return 6
35 fi
36 nslookup $mypublicip | grep "name =" | awk -F ' ' '{print $4}' | sed 's/com\./com/' > /etc/hostname
37 myrc=$?
38 if [ $myrc -ne 0 ]

```

```

39 then
40     echo "My public DNS name not defined" >> $LOG
41     return 7
42 fi
43 hostname -F /etc/hostname
44
45 # This script modifies the CVMFS and Frontier scripts on VM startup
46 # to point to the ELB-enabled squid stack for the respective
47 # availability zone
48
49 zone=$(curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone)
50 echo $zone
51 echo $zone >> $LOG
52 addr="http://elb2.$zone.elb.fnaldata.org:3128" #HK> export http_proxy does not like two
... back-slashes, I had to remove them
53
54 # new code by ST and HK ##########
55 export http_proxy=$addr
56 wget http://cvmfs.fnal.gov:8000/cvmfs/cms.cern.ch/.cvmfspublished
57 returnvalue=$?
58 if [ $returnvalue -ne 0 ]
59 then
60     echo "Squid Server is not accessible in $zone, trying us-west-2b" >> $LOG
61
62     if [ $zone != "us-west-2b" ]
63     then
64         uswest2baddr="http://elb2.us-west-2b.elb.fnaldata.org:3128"
65         export http_proxy=$uswest2baddr
66         wget http://cvmfs.fnal.gov:8000/cvmfs/cms.cern.ch/.cvmfspublished
67         returnvalue=$?
68         if [ $returnvalue -ne 0 ]
69         then
70             echo "Squid Server is not accessible at all" >> $LOG
71             return 11
72         else
73             echo "Squid Server is available in us-west-2b" >> $LOG
74             addr="http://elb2.us-west-2b.elb.fnaldata.org:3128"
75         fi

```

```

76
77     else
78         echo "Squid Server is not available even in us-west-2b" >> $LOG
79         return 11
80     fi
81
82 else
83     echo "Squid Server is available in $zone" >> $LOG
84     addr="http:\/\/elb2.$zone.elb.fnaldata.org:3128" #HK> now, sed command below requires two
... back-slashes, I had restore them here.
85 fi
86 # END: new code by HK and ST ##########
87
88
89 # make sure /etc/cvmfs/default.local is in place
90 if [ ! -r /etc/cvmfs/default.local ]
91 then
92     echo "/etc/cvmfs/default.local not found" >> $LOG
93     return 8
94 fi
95 # and modify
96 sed -i -e "s/^(CVMFS_HTTP_PROXY=).*/\1$addr/" /etc/cvmfs/default.local
97
98
99
100 # make sure /usr/bin/cvmfs_config is in place
101 if [ ! -x /usr/bin/cvmfs_config ]
102 then
103     echo "/usr/bin/cvmfs_config not executable" >> $LOG
104     return 9
105 fi
106 # and run it
107 /usr/bin/cvmfs_config reload >> $LOG 2>&1
108
109
110
111 # make sure /etc/cvmfs/SITECONF/local/JobConfig/site-local-config.xml is in place
112 if [ ! -r /etc/cvmfs/SITECONF/T3_US_HEP_Cloud/JobConfig/site-local-config.xml ]

```

```

113 then
114     echo "/etc/cvmfs/SITECONF/T3_US_HEP_Cloud/JobConfig/site-local-config.xml not found" >> $LOG
115     return 10
116 fi
117 # and modify
118 sed -i -e "s/^(<proxy url=\").*/\1\"$addr\"/>/" 
... /etc/cvmfs/SITECONF/T3_US_HEP_Cloud/JobConfig/site-local-config.xml
119
120
121
122 # getting rid of the Fermi-specific crlsquid.fnal.gov and crl-cache.fnal.gov if it's there
123 if [ -r /etc/fetch-crl.conf ]
124 then
125     cp -p /etc/fetch-crl.conf      /etc/fetch-crl.conf.fermisav
126     sed -i -e "s/(http_proxy = \").*/\1$addr/" /etc/fetch-crl.conf
127     grep -v prepend_url /etc/fetch-crl.conf > /etc/fetch-crl.conf.temp
128     cp      /etc/fetch-crl.conf.temp /etc/fetch-crl.conf
129 #    start a fetch-crl now at S27 instead of enabling the fetch-crl-boot
130 #    and throw it into background.
131
132 # commented out by HK/ST in order to make sure fetch-crl does not populate
... /etc/grid-security/certificates/ with *.r0 files
133 #    nohup /usr/sbin/fetch-crl < /dev/null >> $LOG 2>&1 &
134 fi
135
136 return 0
137 }
138
139 stop() {
140
141 rm /var/lock/subsys/hepcloud-init-workernode
142
143 return 0
144 }
145
146 case "$1" in
147     start)
148         start

```

```

149      RETVAL=$?
150  ;;
151 stop)
152   stop
153   RETVAL=$?
154 ;;
155 restart|try-restart|condrestart)
156   ## Stop the service and regardless of whether it was
157   ## running or not, start it again.
158   #
159   ## Note: try-restart is now part of LSB (as of 1.9).
160   ## RH has a similar command named condrestart.
161   start
162   RETVAL=$?
163 ;;
164 reload|force-reload)
165   # It does not support reload
166   RETVAL=3
167 ;;
168 status)
169   echo -n $"Checking for service $prog:"
170   # Return value is slightly different for the status command:
171   # 0 - service up and running
172   # 1 - service dead, but /var/run/ pid file exists
173   # 2 - service dead, but /var/lock/ lock file exists
174   # 3 - service not running (unused)
175   # 4 - service status unknown :-((
176   # 5--199 reserved (5--99 LSB, 100--149 distro, 150--199 appl.)
177   RETVAL=3
178 ;;
179 *)
180   echo "Usage: $0 {start|stop|status|try-restart|condrestart|restart|force-reload|reload}"
181   RETVAL=3
182 ;;
183 esac
184
185 exit $RETVAL
186

```

```

187 -----
188 -----
189 -----
190
191
192 File: ondemandservices/largequery.sh
193 #!/bin/bash
194
195 #This script should be present in the /root directory to work.If you are planning to place to
196 # somewhere else please change the path in the lines 6-8
197
198 zone=$(curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone)
199 echo zone
200
201 addr="http://elb.$zone.elb.fnaldata.org:3128"
202
203 sed -i -e "s/^(CVMFS_HTTP_PROXY=).*/\1$addr/" ..etc/cvmfs/default.local
204
205 export http_proxy=http://elb2.us-west-2b.elb.fnaldata.org:3128
206
207 export STRATUM1URL="http://cvmfs.fnal.gov:8000"
208
209 NUMPROCESSES=25
210
211 WGETSPERITER=100
212
213 let I=0
214
215 #RUNNING="running.`uname -n`"
216
217 #trap "rm -f $RUNNING" 0
218
219 #touch $RUNNING
220
221 let P=0
222
223 while [ $P -lt $NUMPROCESSES ]; do

```

```

224
225     let P=$P+1
226
227     (   let I=$I+1
228
229         echo "Iteration $P:$I"
230
231         time bash -c 'n=0; while [ $n -lt "'$WGETSPERITER'" ]; do wget -qO/dev/null
232         ... "$STRATUM1URL/cvmfs/fermilab.opensciencegrid.org/data/3d/d8c5d91b8a94cd26a734c48188d5bbc223855bP";
233         let n=$n+1; done' ) > /tmp/$P.out 2>&1 &
234
235 done
236
237
238 -----
239 -----
240 -----
241
242
243 File: ondemandservices/smallquery.sh
244 #!/bin/bash
245
246 #This script should be present in the /root directory to work.If you are planning to place to
247 ... somewhere else please change the path in the lines 6-8
248
249 zone=$(curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone)
250
251 echo zone
252
253 addr="http://elb2.$zone.elb.fnaldata.org:3128"
254
255 export http_proxy=$addr
256
257 export STRATUM1URL="http://cvmfs.fnal.gov:80"
258
259 NUMPROCESSES=25

```

```

259
260 WGETSPERITER=25
261
262 REPEATSPERWGET=500
263
264 let I=0
265
266 #RUNNING="running.`uname -n`"
267
268 #trap "rm -f $RUNNING" 0
269
270 #touch $RUNNING
271
272 let P=0
273
274 URL="$STRATUM1URL/cvmfs/fermilab.opensciencegrid.org/.cvmfspublished"
275
276 let U=0
277
278 export URLs=""
279
280 while [ $U -lt $REPEATSPERWGET ]; do
281
282     let U=$U+1
283
284     URLs="$URLS $URL"
285
286 done
287
288 while [ $P -lt $NUMPROCESSES ]; do
289
290     let P=$P+1
291
292 #     while [ -f $RUNNING ]; do
293
294         (   let I=$I+1
295
296             echo "Iteration $P:$I"

```

```
297
298         time bash -c 'n=0; while [ $n -lt '"$WGETSPERITER"' ];
299
300 do wget -q0/dev/null $URLS; let n=$n+1; done' ) > /tmp/$P.out 2>&1 &
301
302         # exit
303
304 #      done &
305
306 done
307
308 wait
309
310 iftop
311
312
313
314
315 -----
316 -----
317 -----
318
319
320 File: ondemandservices/squid-client-addr-change.sh
321 #!/bin/bash
322 #This script should be present in the /root directory to work.If you are planning to place to
323 .. somewhere else please change the path in the lines 6-8
324 zone=$(curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone)
325 echo zone
326 addr="http://elb2.$zone.elb.fnaldata.org:3128"
327 sed -i -e "s/^(CVMFS_HTTP_PROXY=).*/\1$addr/" ..etc/cvmfs/default.local
328 sed -i -e "s/(http_proxy=).*/\1$addr/" ..etc/cvmfsload/smallquery
329 sed -i -e "s/(http_proxy=).*/\1$addr/" ..etc/cvmfsload/largequery
330
331 -----
332 -----
333 -----
```

```

1 File: spotpricehistory/SpotPriceHistory.py
2
3 import boto3
4 import datetime
5 import os.path
6 from boto3.session import Session
7
8 class SpotPriceHistory:
9     """
10         This class is used for getting spot pricing history
11     """
12
13     startTime = ""
14     endTime = ""
15     zone = "us-west-2a"
16     instanceType="m3.medium"
17     os="Linux/UNIX"
18     #historyData = {}
19     #dataList=[]
20     nextToken=""
21     def __init__(self,instanceType,zone):
22         self.instanceType=instanceType
23         self.zone=zone
24         self.historyData={}
25         self.dataList=[]
26         self.filename="Database/"+self.instanceType+"_"+self.zone
27         self.readLastTimeFromDatabase()
28
29     def set_startTime(self,startTime):
30         self.startTime=startTime
31     def set_endTime(self,endTime):
32         self.endTime=endTime
33     def set_zone(self,zone):
34         self.zone=zone
35     def set_instanceType(self,instanceType):
36         self.instanceType=instanceType
37     def set_os(self,os):
38         self.os=os

```

```

39
40
41     def getSpotPriceHistory(self):
42         session = Session(aws_access_key_id='',
43                            aws_secret_access_key='',
44                            region_name=self.zone[:-1])
45         client = session.client('ec2')
46         iterates=0
47         while iterates==0 or self.nextToken!="":
48             temp = client.describe_spot_price_history(
49                 DryRun=False,
50                 StartTime=self.startTime,
51                 EndTime=self.endTime,
52                 InstanceTypes=[self.instanceType],
53                 ProductDescriptions=[self.os],
54                 Filters=[],
55                 AvailabilityZone= self.zone,
56                 MaxResults=1000,
57                 NextToken=self.nextToken
58             )
59             self.dataList.insert(0, temp)
60             #tempDic=self.historyData.copy()
61             #tempDic.update(temp)
62             #self.historyData=tempDic
63             self.nextToken=temp['NextToken']
64             iterates+=1
65     def printHistoryData(self):
66         for dicts in self.dataList:
67             for i in reversed(dicts['SpotPriceHistory']):
68                 print
69                 (i['InstanceType'],i['ProductDescription'],i['SpotPrice'],str(i['Timestamp']),i['
70                 AvailabilityZone'])
71
72     def getCredentials(self):
73         """
74             Get AWS credentials from file
75         """

```

```

75     def writeHistoryData(self):
76         '''
77             Write the historical data into database
78         '''
79         filename=self.filename
80         if not os.path.isfile(filename):
81             f=open(filename,"w")
82             f.write("DateTime Price InstanceType Zone\n")
83             for dicts in self.dataList :
84                 for i in reversed(dicts['SpotPriceHistory']):
85                     f.write(i['Timestamp'].strftime("%Y-%m-%dT%H:%M:%S.%f")+""
86 "...+str(i['SpotPrice'])+" "+str(i['InstanceType'])+" "+str(i['AvailabilityZone'])+"\n")
87             f.close()
88         else:
89             f=open(filename,"a")
90             for dicts in self.dataList :
91                 for i in reversed(dicts['SpotPriceHistory']):
92                     for t in i['Timestamp'].timetuple():
93                         print t
94                     for tt in self.startTime.timetuple():
95                         print tt
96                         if i['Timestamp'].replace(tzinfo=None)>self.startTime:
97                             f.write(i['Timestamp'].strftime("%Y-%m-%dT%H:%M:%S.%f")+""
98 "...+str(i['SpotPrice'])+" "+str(i['InstanceType'])+" "+str(i['AvailabilityZone'])+"\n")
99                         f.close()
100
101
102     def readLastTimeFromDatabase(self):
103         '''
104             read last time stamp from the Database, and set start time and end time
105         '''
106         self.endTime=datetime.datetime.utcnow()
107         filename = self.filename
108         if not os.path.isfile(filename):
109             print ("File does not exist! Start from 90 days ago!")
110             self.startTime=self.endTime-datetime.timedelta(days=90)
111
112

```

```

111         with open(filename,"r") as f:
112             for lines in f:
113                 pass
114             last=lines
115             #content=f.read().splitlines()
116             #tempStr=content[len(content)-2].split(" ")
117             tempStr=last.split(" ")
118             print "Last time stamp: " + tempStr[0]
119             self.startTime=datetime.datetime.strptime(tempStr[0],"%Y-%m-%dT%H:%M:%S.%f")
120             f.close()
121             #self.startTime=self.startTime.replace(tzinfo=None)
122             #for tt in self.startTime.timetuple():
123             #    print tt
124
125
126 vCPUs = {
127     'c3.large':2,
128     'c3.xlarge':4,
129     'c3.2xlarge':8,
130     'c3.4xlarge':16,
131     'c3.8xlarge':32,
132     'm3.medium':1,
133     'm3.large':2,
134     'm3.xlarge':4,
135     'm3.2xlarge':8,
136 }
137
138 ecu = {
139     'c3.large':7,
140     'c3.xlarge':14,
141     'c3.2xlarge':28,
142     'c3.4xlarge':55,
143     'c3.8xlarge':108,
144     'm3.medium':3,
145     'm3.large':6.5,
146     'm3.xlarge':13,
147     'm3.2xlarge':26,
148 }
```

```
149
150 std_prices = {
151     'c3.large' :0.105,
152     'c3.xlarge':0.210,
153     'c3.2xlarge':0.420,
154     'c3.4xlarge':0.840,
155     'c3.8xlarge':1.680,
156     'm3.medium' :0.070,
157     'm3.large'  :0.140,
158     'm3.xlarge' :0.280,
159     'm3.2xlarge':0.560
160 }
161
162 -----
163 -----
164 -----
165
166
167 File: spotpricehistory/updateDatabase.py
168 from SpotPriceHistory import SpotPriceHistory
169 import datetime
170 import os.path
171 #from analysis import simulation
172
173 instances=["m3.2xlarge","c3.2xlarge","m3.xlarge","c3.xlarge","m3.medium","m3.large","m3.xlarge"
174 ...,"c3.large","c3.4xlarge","c3.8xlarge"]
175 zone=[{"us-east-1b",
176 ...,"us-east-1c","us-east-1d","us-east-1e","us-west-1a","us-west-1c","us-west-2a",
177 ...,"us-west-2b","us-west-2c",]
178
179 if not os.path.exists("Database"):
180     os.mkdir("Database")
181 if not os.path.exists("Histogram"):
182     os.mkdir("Histogram")
183
184 for i in instances:
185     for z in zone:
```

```
184     awsPrice=SpotPriceHistory(i,z)
185     awsPrice.getSpotPriceHistory()
186     awsPrice.writeHistoryData()
187 #         analyze=simulation(i,z)
188 #         analyze.writeHistogram()
189     print i + " in " + z +" finishes!"
190
191 -----
192 -----
193 -----
```

```

1 File: imagemanagement/step2/config.py
2 import os
3
4 import ini_handler
5
6 from errors import ConfigError
7
8 from simple_logging import Logger
9 from simple_logging import FileWriter
10 from simple_logging import SyslogWriter
11 from simple_logging import ConsoleWriter
12
13 from contextualization_types import CONTEXT_TYPE_EC2
14 from contextualization_types import CONTEXT_TYPE_NIMBUS
15 from contextualization_types import CONTEXT_TYPE_OPENNEBULA
16
17 class Config(object):
18     valid_context_types = [CONTEXT_TYPE_EC2, CONTEXT_TYPE_NIMBUS, CONTEXT_TYPE_OPENNEBULA]
19
20     def __init__(self, config_ini="/etc/glideinwms/glidein-pilot.ini"):
21         if not os.path.exists(config_ini):
22             raise ConfigError("%s does not exist" % config_ini)
23
24         self.ini = ini_handler.Ini(config_ini)
25
26         self.default_max_lifetime = self.ini.get("DEFAULT", "default_max_lifetime", "172800")
27         # 48 hours
28         self.max_lifetime = self.default_max_lifetime # can be overridden
29         self.disable_shutdown = self.ini.getBoolean("DEFAULT", "disable_shutdown", False)
30         self.max_script_runtime = self.ini.get("DEFAULT", "max_script_runtime", "60")
31
32         self.pre_script_dir = self.ini.get("DIRECTORIES", "pre_script_dir",
33         "/usr/libexec/glideinwms_pilot/PRE")
34         self.post_script_dir = self.ini.get("DIRECTORIES", "post_script_dir",
35         "/usr/libexec/glideinwms_pilot/POST")
36
37         # home directory is created by the rpm
38         self.home_dir = "/home/glidein_pilot"

```

```

36         self.glidein_user = "glidein_pilot"
37 #HK
38         self.scratch_dir = "/home/scratchgwms"
39
40         # glidein_startup.sh specific attributes
41         self.factory_url = ""
42         self.pilot_args = ""
43         self.proxy_file = ""
44         self.pilot_args = ""
45
46     def setup(self):
47         self.setup_logging()
48         self.setup_pilot_files()
49         self.setup_contextualization()
50
51     def setup_pilot_files(self):
52         self.ini_file = "%s/glidein_userdata" % self.home_dir
53         self.userdata_file = "%s/userdata_file" % self.home_dir
54         self.log.log_info("Default ini file: %s" % self.ini_file)
55         self.log.log_info("Default userdata file: %s" % self.userdata_file)
56
57     def setup_contextualization(self):
58         self.contextualization_type = self.ini.get("DEFAULT", "contextualize_protocol")
59         self.log.log_info("Contextualization Type identified as: %s" %
60         self.contextualization_type)
61         if self.contextualization_type in Config.valid_context_types:
62             if self.contextualization_type == CONTEXT_TYPE_EC2:
63                 self.ec2_url = self.ini.get("DEFAULT", "ec2_url")
64             elif self.contextualization_type == CONTEXT_TYPE_NIMBUS:
65                 self.nimbus_url_file = self.ini.get("DEFAULT", "nimbus_url_file")
66             elif self.contextualization_type == CONTEXT_TYPE_OPENNEBULA:
67                 self.one_user_data_file = self.ini.get("DEFAULT", "one_user_data_file")
68             else:
69                 raise ConfigError("configured context type not valid")
70
71     def setup_logging(self):
72         log_writer = None
73         log_writer_class = self.ini.get("DEFAULT", "logger_class", None)

```

```

73     if log_writer_class:
74         if log_writer_class == "SyslogWriter":
75             facility = self.ini.get("DEFAULT", "syslog_facility", None)
76             priority = self.ini.get("DEFAULT", "syslog_priority", None)
77             log_writer = SyslogWriter(facility=facility, priority=priority)
78         elif log_writer_class == "ConsoleWriter":
79             output = self.ini.get("DEFAULT", "console_output", "stdout")
80             log_writer = ConsoleWriter(output=output)
81         else:
82             log_writer = FileWriter(self.home_dir)
83     else:
84         #log_writer = FileWriter(self.home_dir)
85         log_writer = FileWriter('/var/log/glideinwms-pilot')
86     self.log = Logger(log_writer)
87     self.log.log_info("Pilot Launcher started...")
88
89 def export_custom_env(self):
90     """
91     @returns: string containing the shell (sh, bash, etc) directives to
92             export the environment variables
93     """
94     environment = ""
95     try:
96         env = self.get_custom_env()
97         for option in env:
98             environment += "export %s=%s; " % (option, env[option])
99     except:
100         # pylint: disable=W0702
101         pass
102     return environment
103
104 def get_custom_env(self):
105     """
106     Returns a dictionary of the parent process environment plus the custom
107     environment variables defined in the pilot config file.
108
109     NOTE: All custom environment variable names will be upper cased. The
110           values for the custom environment variables will not be modified.

```

```

111
112     @returns: dictionary containing the desired process environment
113     """
114     environment = {}
115     # inherit the parent process environment
116     for var in os.environ.keys():
117         environment[var] = os.environ[var]
118
119     try:
120         # add in the custom environment
121         for option in self.cp.ini.options("CUSTOM_ENVIRONMENT"):
122             environment[str(option).upper()] = self.ini.get("CUSTOM_ENVIRONMENT", option)
123     except:
124         # pylint: disable=W0702
125         pass
126
127     # Add in the pilot proxy
128     environment["X509_USER_PROXY"] = self.proxy_file
129     environment["HOME"] = self.home_dir
130     environment["LOGNAME"] = self.glidein_user
131
132     environment["SCRATCH"] = self.scratch_dir
133     return environment
134
135 -----
136 -----
137 -----
138
139
140 File: imagemanagement/step2/hkpilot.sh
141 #!/bin/bash
142 export http_proxy="http://131.225.148.121:3128"
143 export https_proxy="http://131.225.148.121:3128"
144
145 mkdir -p /etc/puppet/modules
146 #puppet module install desalvo-cvmfs
147
148 cat << EOF | puppet apply

```

```

149 class cvmfs {
150 }
151 class cvmfs::client (
152     \$repositories = 'sft.cern.ch',
153     \$quota_limit = 30000,
154     \$http_proxy = undef,
155 ) inherits cvmfs {
156
157     package { cvmfs: ensure => installed, require => Package['osg-release'] }
158     package { cvmfs-config-osg: ensure => installed, require => Package['osg-release'] }
159     if (!defined(Package["fuse"])) {
160         package { fuse: ensure => latest }
161     }
162
163     if (!\$http_proxy) {
164         \$default_http_proxy = 'DIRECT'
165     } else {
166         \$default_http_proxy = '\${http_proxy};DIRECT'
167     }
168
169     file {
170         '/etc/cvmfs/default.local':
171             owner => root, group => root, mode => 644,
172             content => inline_template("CVMFS_REPOSITORIES=<%= repositories -%>
... \nCVMFS_QUOTA_LIMIT=<%= quota_limit -%> \nCVMFS_HTTP_PROXY=\"<%= default_http_proxy -%>\" \n
... "),
173             notify => Exec['cvmfs setup']
174     }
175
176     exec { 'cvmfs reload':
177         path => [ '/bin', '/usr/bin' ],
178         command => 'cvmfs_config reload',
179         timeout => 0,
180         refreshonly => true,
181         require => [Package['cvmfs'],Package['cvmfs-config-osg']]
182     }
183
184

```

```

185     exec { 'cvmfs setup':
186         path => [ '/bin', '/usr/bin' ],
187         command => 'cvmfs_config setup',
188         timeout => 0,
189         require => [Package['cvmfs'],Package['cvmfs-config-osg']]
190     }
191 }
192 group { 'cvmfs-gid':
193     name    => 'cvmfs',
194     gid    => '9125',
195     ensure  => 'present',
196 }
197 user { 'cvmfs-uid':
198     name    => 'cvmfs',
199     uid    => '46084',
200     gid    => '9125',
201     home   => '/var/lib/cvmfs',
202     comment => 'CernVM File System',
203     shell   => '/sbin/nologin',
204     ensure  => 'present',
205     require  => Group['cvmfs-gid'],
206 }
207
208 package { 'yum-conf-epel':
209     ensure  => 'installed',
210 }
211
212 package { 'yum-plugin-priorities':
213     ensure  => 'installed',
214 }
215
216 package { 'osg-release':
217     provider => 'rpm',
218     source   => 'http://repo.grid.iu.edu/osg/3.2/osg-3.2-el6-release-latest.rpm',
219 }
220
221 yumrepo { 'epel':
222

```

```

223     enabled  => 1,
224     require  => Package['yum-conf-epel'],
225   }
226
227 package { 'osg-ca-certs':
228   ensure  => 'installed',
229   require => Package['osg-release']
230 }
231
232 package { 'osg-oasis':
233   ensure  => 'installed',
234   require => [ Package['osg-release'], User['cvmfs-uid'] ],
235 }
236
237 package { 'osg-wn-client':
238   ensure  => 'installed',
239   require => Package['osg-release']
240 }
241
242 package { 'fetch-crl':
243   ensure  => 'installed',
244   require => Package['yum-conf-epel']
245 }
246
247 service {'fetch-crl-boot':
248   ensure  => 'running',
249   enable   => 'false',
250   require  => [ File['/etc/fetch-crl.conf'], Package['osg-wn-client'], ]
251 }
252
253 service {'fetch-crl-cron':
254   ensure  => 'running',
255   enable   => 'false',
256   require  => [ File['/etc/fetch-crl.conf'], Package['osg-wn-client'], ]
257 }
258
259 file { '/etc/fetch-crl.conf': #Not sure this is not the default already....
260   ensure  => file,

```

```

261   mode      => '644',
262   owner     => 'root',
263   group     => 'root',
264   content   => '#'
265 ## PUPPET MAINTAINED file for fetch-crl
266 ##
267 infodir = /etc/grid-security/certificates
268 agingtolerance = 24
269 nosymlinks
270 nowarnings
271 noerrors
272 stateless
273 logmode = syslog
274 syslogfacility = daemon
275 http_proxy = http://crlsquid.fnal.gov:3128
276 prepend_url = http://crl-cache.fnal.gov/certificates/@ANCHORNAME@.r@R@
277 ,
278   require => Package['fetch-crl']
279 }
280
281 class { 'cvmfs::client':
282   repositories =>
283   'oasis.opensciencegrid.org,atlas.cern.ch,atlas-condb.cern.ch,cms.cern.ch,lhcbo.cern.ch,nova.
284   fnaldata.gov',
285   quota_limit  => 4000,
286   http_proxy   => 'squid.fnal.gov:3128',
287   require      => Package['osg-oasis'],
288 }
289
290 file { '/etc/cvmfs/keys/fnaldata.gov.pub':
291   ensure  => 'file',
292   content => "-----BEGIN PUBLIC
293 KEY-----\nMIIBIjANBkgkhkiG9w0BAQEAA0CAQ8AMIIIBCgKCAQEAngbplwxU8YgAoTcF51cr\
294 nmt28kSwUMbn8fj4kK0kwTHlC1Hsh6sWbrTl/iZhWJDFPyVqnIyDGEHDUKHSt9XE\
295 nWgdoz6VXoHbiVNg3xLLE30RCycQumeb/wXf0juvL77LGvMmspmBB/2W/0Gprw7Kr\
296 n4dWd9LHzm90Lx1WEJpGLsdPzAvhqgAx1TYaIuB9pG/M+Rh5ep4ZRQArpvrl6yxL\
297 njBWlPavDrsBnJj0wqB27G0inBsKro5d8qeGYmNyWC3cKP5aVZ9IoL/7XI7RLxEtp\
298 nfrhMp1l9AmzDQpqAfGqPS1G5ziR0PB7evSJBrP08An2o+w15I0CF89mjm9I8IsVW\n4QIDAQAB\n-----END PUBLIC

```

```

290... KEY-----\n",
291      require => Class['cvmfs::client'],
292    }
293
294    file { '/etc/cvmfs/domain.d/fnaldata.gov.conf':
295      ensure => 'file',
296      content =>
297        "...CVMFS_PUBLIC_KEY=/etc/cvmfs/keys/fnaldata.gov.pub\nCVMFS_SERVER_URL='http://cvmfss1data.fnal.
298        gov:8000/cvmfs/@fqrn@;http://hcc-cvmfs.unl.edu:8000/cvmfs/@fqrn@\nCVMFS_HTTP_PROXY=DIRECT\
299        \nCVMFS_ALIEN_CACHE=/pnfs/fs/usr/cvmfs/fermicloud-alien-cache\nCVMFS_QUOTA_LIMIT=-1\
300        \nCVMFS_SHARED_CACHE=no\n",
301      require => Class['cvmfs::client'],
302    }
303
304    file { ['/pnfs', '/pnfs/fs', '/pnfs/fs/usr', '/pnfs/fs/usr/cvmfs']:
305      ensure => 'directory',
306    }
307
308    mount { '/pnfs/fs/usr/cvmfs':
309      device => 'pnfs-stoken:/pnfs/fs/usr/cvmfs',
310      fstype => 'nfs4',
311      ensure => 'mounted',
312      options => 'minorversion=1',
313      atboot => 'true',
314      require => File['/pnfs/fs/usr/cvmfs'],
315    }
316
317    yumrepo {'glideinwms':
318      descr    => 'Glideinwms repository 6 - i\$basearch',
319      baseurl  => 'http://fermigrid.fnal.gov/files/glideinwms/prod/6/\$basearch/',
320      enabled   => 1,
321      gpgcheck  => 0,
322      'glideinwms-dev':
323        descr    => 'Glideinwms dev repository 6 - \$basearch',
324        baseurl  => 'http://fermigrid.fnal.gov/files/glideinwms/dev/6/\$basearch/',
325        enabled   => 0,
326        gpgcheck  => 0,
327    }

```

```

324
325 package { 'glideinwms-vm-one':
326   ensure  => 'installed',
327   require => Yumrepo['glideinwms'],
328   notify => Exec['/usr/sbin/ntpdate'],
329 }
330
331 service {'glideinwms-pilot':
332   ensure  => 'running',
333   enable   => 'true',
334   require  => Exec['hktempoverwrite'],
335 }
336
337 exec { 'hktempoverwrite':
338   command => '/bin/mount -t iso9660 /dev/sr0 /mnt;/bin/cp -f /mnt/pilot-launcher
339   ... /usr/sbin/pilot-launcher; /bin/cp -f /mnt/mount_ephemeral
340   ... /usr/libexec/glideinwms_pilot/PRE/mount_ephemeral; /bin/cp -f /mnt/config.py
341   ... /usr/lib/python2.6/site-packages/glideinwms_pilot/config.py',
342   require  => Package['glideinwms-vm-one'],
343 }
344
345 exec {'/usr/sbin/ntpdate':
346   refreshonly => true,
347   command => '/usr/sbin/ntpdate -s 131.225.82.122',
348   require  => [Service['glideinwms-pilot']]]
349
350
351
352 -----
353 -----
354 -----
355
356
357 File: imagemanagement/step2/internal.py
358 #!/usr/bin/python

```

```

359 import time
360 import subprocess
361
362 def main():
363     command0 = "service glideinwms-pilot status"
364     with open('/tmp/hktesting.log', 'a') as my_log:
365         retcode=subprocess.call(command0, shell=True, stdout=my_log)
366         while retcode != 0:
367             retcode=subprocess.call(command0, shell=True, stdout=my_log)
368             print "retcode = ", retcode
369             time.sleep(1)
370
371     with open('/tmp/hktesting.log', 'a') as my_log:
372         my_log.write("HK glidein running, now trying to stop it")
373
374
375     command1 = "service glideinwms-pilot stop"
376     with open('/tmp/hktesting.log', 'a') as my_log:
377         retcode = subprocess.call(command1, shell=True, stdout=my_log)
378
379
380     with open('/tmp/hktesting.log', 'a') as my_log:
381         retcode=subprocess.call(command0, shell=True, stdout=my_log)
382         print "retcode = ", retcode
383         while retcode != 3:
384             retcode=subprocess.call(command0, shell=True, stdout=my_log)
385             print "retcode = ", retcode
386             time.sleep(1)
387     with open('/tmp/hktesting.log', 'a') as my_log:
388         my_log.write("HK glidein stopped")
389
390     command2 = "cat /dev/null > /var/log/glideinwms-pilot/pilot_launcher.log"
391     command3 = "rm -f /tmp/ephemeral_storage.log"
392     command4 = "umount /home && mv /home.orig/* /home && rmdir /home.orig"
393
394
395     with open('/tmp/hktesting.log', 'a') as my_log:
396         retcode2 = subprocess.call(command2, shell=True, stdout=my_log)
397         print "retcode 2 = ", retcode2

```

```

397         time.sleep(2)
398         retcode3 = subprocess.call(command3, shell=True, stdout=my_log)
399         print "retcode 3 = ", retcode3
400         time.sleep(2)
401         retcode4 = subprocess.call(command4, shell=True, stdout=my_log)
402         print "retcode 4 = ", retcode4
403         time.sleep(2)
404
405
406         with open('/tmp/hktesting.log', 'a') as my_log:
407             my_log.write( "all jobs done and now shutting down now" )
408
409         command5 = "shutdown -h now"
410         with open('/tmp/hktesting.log', 'a') as my_log:
411             retcode = subprocess.call(command5, shell=True, stdout=my_log)
412
413
414 if __name__ == "__main__":
415     main()
416
417 -----
418 -----
419 -----
420
421
422 File: imagemanagement/step2/internal.sh
423 #!/bin/bash
424 nohup /mnt/internal.py > /dev/null &
425
426 -----
427 -----
428 -----
429
430
431 File: imagemanagement/step2/mount_ephemeral
432 #!/bin/bash
433 #

```

```

435 # mount_ephemeral - Attempts to mount ephemeral storage
436 #
437
438
439
440 LOGFILE=/tmp/ephemeral_storage.log
441 VIRTUAL_DISKS="xvdf xvdb vda3 vdb sda2 sdb sdbe"
442
443 echo "Attempting to mount ephemeral storage" | tee $LOGFILE
444
445 if [ ! -d /home/scratchgwms ]; then
446     mkdir -v /home/scratchgwms 2>&1 | tee --append $LOGFILE
447 fi
448
449 for VD in $VIRTUAL_DISKS ; do
450     echo "Checking /dev/$VD ..." | tee --append $LOGFILE
451     d=`date`
452     fdisk -l /dev/$VD 2>/dev/null | grep Disk >> $LOGFILE
453     if [ $? -eq 0 ]; then
454         echo "Virtual disk seen at /dev/$VD: $d" | tee $LOGFILE
455
456 #HK hack begin
457     string1=`grep scratchgwms /etc/mtab`
458     if [ -n "$string1" ]; then
459         echo "/home/scratchgwms is currently mounted" | tee --append $LOGFILE
460         (cd /home/scratchgwms && rm -rf *) 2>&1 | tee --append $LOGFILE
461         umount /home/scratchgwms 2>&1 | tee --append $LOGFILE
462     fi
463
464     if [ ! -d /home/scratchgwms ]; then
465         mkdir -v /home/scratchgwms 2>&1 | tee --append $LOGFILE
466     fi
467
468     mount /dev/$VD /home/scratchgwms 2>&1 | tee --append $LOGFILE
469 #HK hack end
470
471     d=`date`
472     echo "Done with /dev/$VD: $d" | tee --append $LOGFILE

```

```

473         break
474     else
475         echo "No virtual disk seen at /dev/$VD: $d" | tee --append $LOGFILE
476     fi
477 done
478 d=`date`
479 echo "Done: $d" | tee --append $LOGFILE
480
481 -----
482 -----
483 -----
484
485
486 File: imagemanagement/step2/pilot-launcher
487 #!/usr/bin/python
488
489 import os
490 import subprocess
491 import signal
492 import urllib
493 from optparse import OptionParser
494
495 from glideinwms_pilot.errors import PilotError
496 from glideinwms_pilot.errors import TimeoutError
497 from glideinwms_pilot.errors import ConfigError
498 from glideinwms_pilot.errors import ScriptError
499
500 from glideinwms_pilot.vm_utils import chown
501 from glideinwms_pilot.vm_utils import chmod
502 from glideinwms_pilot.vm_utils import cd
503 from glideinwms_pilot.vm_utils import drop_privs
504 from glideinwms_pilot.vm_utils import shutdown_vm
505 from glideinwms_pilot.vm_utils import daemonize
506 from glideinwms_pilot.vm_utils import ls_files_sorted
507 from glideinwms_pilot.vm_utils import sleep
508
509 from glideinwms_pilot.user_data import GlideinWMSUserData
510

```

```

511 from glideinwms_pilot.config import Config
512
513 from glideinwms_pilot.process_handling import execute_cmd
514
515 def retrieve.glidein_startup(config):
516     try:
517         url = "%s/glidein_startup.sh" % config.factory_url
518         script = "%s/glidein_startup.sh" % config.home_dir
519         script, _ = urllib.urlretrieve(url, script)
520     except Exception, ex:
521         raise PilotError("Error retrieving glidein_startup.sh: %s\n" % str(ex))
522
523 def run_scripts(directory, log_writer, max_script_runtime=60, arguments=[]):
524     try:
525         script_list = ls_files_sorted(directory)
526     except Exception, e:
527         message = "An Error has occurred retrieving scripts: %s" % str(e)
528         raise ScriptError(message)
529
530     for script in script_list:
531         try:
532             cmd = "%s/%s" % (directory, script)
533             log_writer.log_info("Executing script %s" % cmd)
534             exit_code = execute_cmd(cmd, max_script_runtime, log_writer,
535                                     arguments, os.environ)
536             log_writer.log_info("Executing script %s ... DONE" % cmd)
537             # have to mod 256 because on some systems, instead of
538             # returning 0 on success, 256 is returned
539             if not int(exit_code) % 256 == 0:
540                 message = "The script (%s) has exited with Exit Code: %s" % (cmd,
541 ... str(exit_code))
542                 log_writer.log_err(message)
543             except Exception, e:
544                 message = "An Error has occurred attempting to run script: %s" \
545                           "\n\nError: %s" % (cmd, str(e))
546             log_writer.log_err(message)
547

```

```

548 def main():
549     """
550     Perform all the work necessary to launch a glideinWMS pilot which will
551     attempt to connect back to the user pool.
552
553     1) daemonize this script. This script is launched via the *nix service
554     mechanisms. We don't want to make it wait forever and we don't
555     want it to be attached to a console.
556     2) Get the user data that was passed to the AMI - Currently it is a
557     tarball.
558     3) untar the tarball. The tarball will contain a proxy, the
559     glidein_startup.sh script and an ini file containing all the extra
560     information needed
561     4) read the ini file
562     5) get the arguments for the glidein_startup.sh script
563     6) create an environment string to pass with final command
564     7) launch the glidein pilot with the appropriate environment
565     """
566
567     usage = "usage: %prog [options] [Site FQDN]"
568     parser = OptionParser(usage=usage)
569     parser.add_option("-d", "--disable-daemon", action="store_true",
570                       dest="disable_daemon", default=False,
571                       help="Disable the daemon functionality and run in \"\
572                             terminal")
573
574     # if the directory, etc/glideinwms, does not exist lets assume that the
575     # config file is in the same directory. We do this now so that this service
576     # can be installed into CMF. This is a nasty hack that isn't portable in
577     # the future, but is being done so that we can move forward. We assume
578     # OpenStack since that is the current whim at CERN.
579     init_config_directory = "/etc/glideinwms"
580     if not os.path.exists(init_config_directory):
581         init_config_directory = os.path.dirname(os.path.abspath(__file__))
582     parser.add_option("-c", "--config-file", dest="config_file",
583                       default="%s/glidein-pilot.ini" % init_config_directory,
584                       help="Specify a custom config file")
585

```

```

586     parser.add_option("-p", "--pid-file", dest="pid_file",
587                         default="/tmp/pilot.pid", help="Specify the pidfile")
588
589
590     (options, args) = parser.parse_args()
591
592
593     if options.disable_daemon:
594         print "disable daemon call"
595     else:
596         daemonize(options.pid_file)
597
598     # If config fails, we need to write error to console if available
599     try:
600         config = Config(options.config_file)
601         config.setup()
602         try:
603             # Change to the working directory -- GUARANTEE A KNOWN start dir
604             config.log.log_info('Changing to: %s' % config.home_dir)
605             cd(config.home_dir)
606             config.log.log_info('Now in: %s' % config.home_dir)
607
608             # Run PRE scripts here
609             message = "Running PRE Scripts in %s" % config.pre_script_dir
610             config.log.log_info(message)
611             run_scripts(config.pre_script_dir, config.log,
612                         float(config.max_script_runtime))
613             message = "Running PRE Scripts in ... DONE"
614             config.log.log_info(message)
615
616             # Change to the working directory AGAIN -- IMPORTANT
617             # Needed since PRE scripts may move the home area to a
618             # bigger partition
619             config.log.log_info('Changing to: %s' % config.home_dir)
620             cd(config.home_dir)
621             config.log.log_info('Now in: %s' % config.home_dir)
622
623             # get and extract the user data - should be a tar file

```

```

624         config.log.log_info("Retrieving and extracting user data")
625         userdata = GlideinWMSUserData(config)
626         userdata.extract_user_data()
627
628         # Change the ownership of files in ~glidein_pilot before
629         # dropping privileges
630         chown('%s.%s' % (config.glidein_user, config.glidein_user),
631               config.home_dir)
632     # HK
633         chown('%s.%s' % (config.glidein_user, config.glidein_user),
634               config.scratch_dir)
635     # HK-end
636
637         # drop privileges to the glidein user
638         config.log.log_info("Dropping privs to %s" % config.glidein_user)
639         drop_privs(config.glidein_user)
640
641         # get the glidein_startup.sh script
642         config.log.log_info("Retrieving glidein_startup.sh")
643         retrieve.glidein_startup(config)
644         chmod(0755, "%s/glidein_startup.sh" % config.home_dir)
645
646         # configure pilot launch environment
647         config.log.log_info("Configuring pilot environment...")
648         config.log.log_info("    Username: %s" % config.glidein_user)
649
650         pilot_env = config.get_custom_env()
651         config.log.log_info("    Environment: %s" % str(pilot_env))
652
653         pilot_args = config.pilot_args.split()
654         pilot_args.insert(0, "glidein_startup.sh")
655         config.log.log_info("Pilot arguments: %s" % str(pilot_args))
656
657         # launch the pilot
658         # The pilot will only be allowed to run for config.max_lifetime
659         # seconds before being terminated
660         glidein_startup = "%s/glidein_startup.sh" % config.home_dir
661         config.log.log_info("Launching Pilot (%s)... "% glidein_startup)

```

```

662         _ = execute_cmd(glidein_startup,
663                         float(config.max_lifetime), config.log,
664                         pilot_args, pilot_env)
665     except ScriptError, ex:
666         message = "An Error has occured: %s" % str(ex)
667         config.log.log_err(message)
668     except PilotError, ex:
669         message = "A PilotError has occured: %s" % str(ex)
670         config.log.log_err(message)
671     except Exception, ex:
672         config.log.log_err("Error launching pilot: %s" % str(ex))
673     except ConfigError, ex:
674         config.disable_shutdown = False
675
676     try:
677         try:
678             message = "Running Post Scripts in %s" % config.post_script_dir
679             config.log.log_info(message)
680         except:
681             # If a config error occurred originally, then logging isn't
682             # available. This is probably what brought us to this point
683             pass
684
685         # Always run POST Scripts
686         run_scripts(config.post_script_dir, config.log,
687                     float(config.max_script_runtime))
688
689         try:
690             message = "Running Post Scripts ... DONE."
691             config.log.log_info(message)
692         except:
693             # If a config error occurred originally, then logging isn't
694             # available. This is probably what brought us to this point
695             pass
696     except ScriptError, ex:
697         try:
698             message = "An Error has occured: %s" % str(ex)
699             config.log.log_err(message)

```

```

700     except:
701         # If a config error occurred originally, then logging isn't
702         # available. This is probably what brought us to this point
703         pass
704
705     # No logging is available if the config.setup call errors, so don't try
706     if config.disable_shutdown:
707         print "shutdown disabled"
708     else:
709         ten_minutes = 600 # seconds
710         sleep(ten_minutes)
711         shutdown_vm(options.pid_file)
712
713 if __name__ == "__main__":
714     main()
715
716 -----
717 -----
718 -----
719
720
721 File: imagemanagement/step2/README
722 in fclheadgpvm01.fnal.gov, the following is there..
723
724 /etc/cron.d/step2_imaging
725 0 7 * * 1-5 root /opt/gcso/opennebula/imaging/step2_imaging_environ.sh
726
727 /opt/gcso/opennebula/imaging/step2_imaging_environ.sh
728
729
730 /opt/gcso/opennebula/imaging/step2_imaging_external.py
731
732 onetemplate list oneadmin
733 185 oneadmin      oneadmin      make_prvm          09/02 12:53:39
734
735 /cloud/images/OpenNebula/scripts/one4.x/contextualization/hkpilot.sh
736 /cloud/images/OpenNebula/scripts/one4.x/contextualization/internal.sh
737 /cloud/images/OpenNebula/scripts/one4.x/contextualization/internal.py

```

```

738
739 /cloud/images/OpenNebula/scripts/one4.x/contextualization/pilot-launcher
740 /cloud/images/OpenNebula/scripts/one4.x/contextualization/mount_ephemeral
741 /cloud/images/OpenNebula/scripts/one4.x/contextualization/config.py
742 The above 3 files will not be needed when I have closed the ticket 10389 to generate a new
.. glideinwms-vm-core 1.0.6 RPM
743
744 INIT_SCRIPTS="init.sh credentials.sh kerberos.sh hkpilot.sh"
745
746 IMAGE="SLF6Vanilla",
747
748 oneimage list oneadmin
749   ID USER      GROUP      NAME          DATASTORE      SIZE TYPE PER STAT RVMS
750   4 oneadmin  oneadmin  SLF6Vanilla  cloud_imag    256G OS   No used  197
751
752
753
754 /cloud/images/OpenNebula/scripts/one4.x/contextualization/hkpilot.sh  is the puppet apply
755
756   exec { 'hktempoverwrite':
757     command => '
758 /bin/mount -t iso9660 /dev/sr0 /mnt;
759 /bin/cp -f /mnt/pilot-launcher /usr/sbin/pilot-launcher;
760 /bin/cp -f /mnt/mount_ephemeral /usr/libexec/glideinwms_pilot/PRE/mount_ephemeral;
761 /bin/cp -f /mnt/config.py      /usr/lib/python2.6/site-packages/glideinwms_pilot/config.py
762 ',
763
764
765
766
767
768 /cloud/images/OpenNebula/scripts/one4.x/contextualization/glidein_startup.sh  is not needed
.. any more.
769
770
771
772
773

```

```

774 -----
775 -----
776 -----
777 -----
778
779
780 File: imagemanagement/step2/step2_imaging_external.py
781 #!/usr/bin/env python
782 import string
783 import subprocess
784 import logging
785 import textwrap
786 import sys
787 import socket
788 import datetime
789 import time
790 import os
791 import time
792 try:
793     import xml.etree.cElementTree as ET
794 except:
795     import xml.etree.ElementTree as ET
796
797
798 def main():
799     my_env = os.environ.copy()
800
801     logging.basicConfig(level=logging.DEBUG,
802                         format='%(asctime)s %(levelname)-8s %(message)s',
803                         datefmt='%b %d %H:%M:%S',
804                         filename='/tmp/step2_image_convert.log')
805
806 #####
807 #####
808     subprocess.Popen( ["onetemplate instantiate make_prvm --name 'step2-AWS' "], env=my_env,
..     stdout=subprocess.PIPE, shell=True )
809
810 # need to pause a bit here after onetemplate instantiate before we will be able to do onevm

```

```

810.. show command
811     time.sleep(60)
812
813
814     getxml = subprocess.Popen( ["onevm show 'step2-AWS' --xml"], env=my_env,
815     .. stdout=subprocess.PIPE, shell=True )
816     std_out_xml, std_err_xml = getxml.communicate()
817     root = ET.fromstring( std_out_xml )
818     vm_id      = root.find('ID').text
819     vm_state_text = root.find('STATE').text
820     vm_state_int = int(vm_state_text)
821
822     logging.info('vm state    = %s' % vm_state_text)
823     logging.info('vm id       = %s' % vm_id)
824
825     while vm_state_int != 3:
826         logging.info('vm is not running yet, sleep 10 more seconds')
827         time.sleep(5)
828
829     getxml = subprocess.Popen(["onevm show 'step2-AWS' --xml"], env=my_env,
830     .. stdout=subprocess.PIPE, shell=True)
831     std_out_xml, std_err_xml = getxml.communicate()
832     root = ET.fromstring( std_out_xml )
833     vm_state_text = root.find('STATE').text
834     vm_state_int = int(vm_state_text)
835     logging.info('vm state    = %s' % vm_state_text)
836
837
838 #HK> new addition
839     kvcommand = '/cloud/images/OpenNebula/scripts/one4.x/onehostname {vmid}'.format(
840     .. vmid=vm_id )
841     getip = subprocess.Popen( kvcommand, env=my_env, stdout=subprocess.PIPE, shell=True)
842     std_out_ip, std_err_ip = getip.communicate()
843     logging.info( 'vm ip address = %s' % std_out_ip.rstrip('\n') )
844
845     kvcommand2 = 'ssh -l root {vmip} uname -a'.format( vmip=std_out_ip.rstrip('\n') )

```

```

845
846     getkv = subprocess.Popen( kvcommand2, env=my_env, stdout=subprocess.PIPE, shell=True)
847     std_out_pl, std_err_pl = getkv.communicate()
848     pilotresult = std_out_pl.rstrip('\n')
849     logging.info( 'pilot result = %s' % pilotresult )
850
851     while not pilotresult.startswith('Linux'):
852         logging.info('vm does not have network, sleep 10 more seconds')
853         time.sleep(10)
854         getkv = subprocess.Popen( kvcommand2, env=my_env, stdout=subprocess.PIPE, shell=True)
855         std_out_pl, std_err_pl = getkv.communicate()
856         pilotresult = std_out_pl.rstrip('\n')
857         logging.info( 'result inside while = %s' % pilotresult )
858
859     logging.info('Finally vm does have network, now we are getting the kernel version')
860
861     kvcommand3 = 'ssh -l root {vmip} uname -r'.format( vmip=std_out_ip.rstrip('\n') )
862     getkv = subprocess.Popen( kvcommand3, env=my_env, stdout=subprocess.PIPE, shell=True)
863     std_out_kv, std_err_kv = getkv.communicate()
864     kernelresult = std_out_kv.rstrip('\n')
865     logging.info('Kernel result = %s' % kernelresult)
866
867     with open('/tmp/kernelversion.txt', 'w') as hkmylog:
868         hkmylog.write( kernelresult )
869
870 #HK> new addition
871
872     logging.info('vm is running and now we are waiting for it to go to poff by glideinwms
873     .. timeout mechanism')
874     ## the internal script will initiate the shutdown
875     # now we wait until the vm does shut down which is executed by the second internal script
876
877     while vm_state_int != 8:
878         logging.info("the vm is still not down")
879         time.sleep(60)
880
881     getxml = subprocess.Popen(["onevm show 'step2-AWS' --xml"], env=my_env,
882     .. stdout=subprocess.PIPE, shell=True)
883     std_out_xml, std_err_xml = getxml.communicate()

```

```

881     root = ET.fromstring( std_out_xml )
882     vm_state_text = root.find('STATE').text
883     vm_state_int = int(vm_state_text)
884
885
886     logging.info('vm is down now, we are going to copy the image out')
887
888 # now the VM is shutdown by the internal script,
889 # this means the opennebula did not kill the VM, i.e. the VMHost still has the disk.0 file at
890 # VMHost:/var/lib/one/datastores/100/VMID/
891
892 # Now, we copy the disk.0 This is a potential risk, i.e. when /opt area does not have
893 # sufficient space, this script will fail to copy out disk.0 from VMHost.
894     tmp_destination = "/cloud/images/hyunwoo/hkdisk.0"
895     tmp_destination = "/opt/gcso/opennebula/imaging/tmp_step2.img"
896     if os.path.exists(tmp_destination):
897         os.remove(tmp_destination)
898     logging.info('%s is deleted' % tmp_destination)
899
900     scp_value = "scp oneadmin@{hostname}:{}/var/lib/one/datastores/104/{vmid}/disk.0
901 {tmpimage}".format(tmpimage=tmp_destination, hostname=bare_metal_host.text, vmid=vm_id)
902
903     subprocess.check_call(scp_value, shell=True)
904
905     logging.info('copy the image out is complete')
906     time.sleep(10)
907
908 ## We need to find out the image location
909 ## Now, we need to replace the old image with the new image:
910     getxml = subprocess.Popen( ["oneimage show 'SLF6_prvM' --xml"], env=my_env,
911     stdout=subprocess.PIPE, shell=True )
912     std_out_xml, std_err_xml = getxml.communicate()
913     root = ET.fromstring( std_out_xml )
914     image_destination = root.find('SOURCE').text
915
916     get_date = datetime.date.today()
917     mv_command = ("mv {tmpimage} {image}.new &&

```

```

918             " mv -f {image} {image}.{today} && "
919             " mv {image}.new {image} && chown oneadmin:oneadmin "
920             "{image} && chmod 660 {image} ").format (tmpimage=tmp_destination,
921             image=image_destination, today=str(get_date) )
922
923     subprocess.check_call(mv_command, shell=True)
924
925     logging.info('copy the image in the image repository is complete')
926
927 #HK, can we delete the poff VM at the end of this script?
928     subprocess.Popen( ["onevm delete 'step2-AWS' "], env=my_env, stdin=subprocess.PIPE,
929     stdout=subprocess.PIPE, shell=True )
930
931     logging.info('onevm delete step2-AWS is complete')
932
933 if __name__ == "__main__":
934     main()
935
936 -----
937 File: imagemanagement/step3/cloud.cfg
938 #Bare-bone cloud.cfg, add parameters as needed for FermiCloud
939
940 user: root
941
942 #If this is not explicitly false, cloud-init will change things so that root
943 #login via ssh is disabled. Set it false to allow root login via ssh keypair.
944
945 disable_root: false
946
947 #add additional cloud-init output logging
948
949 output: {all: '| tee -a /var/log/cloud-init-output.log'}
950

```

```

951 #Since cloud-init runs at multiple stages of boot, this needs to be set so
952 #it can log in all of them to /var/log/cloud-init.
953
954 syslog_fix_perms: null
955
956 #This is the piece that makes userdata work. You need this to have userdata
957 #scripts be run by cloud-init.
958
959 datasource_list: [Ec2]
960 datasource:
961   Ec2:
962     metadata_urls: ['http://169.254.169.254']
963
964 #modules that run early in boot
965
966 cloud_init_modules:
967   - bootcmd #for running commands during boot. Commands can be defined in cloud-config
968   .. userdata.
969
970 #modules that run after boot
971
972 cloud_config_modules:
973   - runcmd #like bootcmd, but runs after boot. Use this instead of bootcmd for after boot
974   .. processing.
975
976 #modules that run at some point after config is finished
977
978 cloud_final_modules:
979   - scripts-per-once #all of these run scripts at specific events. Like bootcmd, can be
980   .. defined in cloud-config.
981   - scripts-per-boot
982   - scripts-per-instance
983   - scripts-user
984   - phone-home #if defined, can make a post request to a specified url when done booting
985   - final-message #if defined, can write a specified message to the log
986   - power-state-change #if defined, can trigger stuff based on power state changes
987
988 system_info:
```

```

986   distro: rhel
987
988 # vim:syntax=yaml
989 -----
990 -----
991 -----
992
993
994 File: imagemanagement/step3/cms.cern.ch.local
995 export CMS_LOCAL_SITE="/etc/cvmfs/SITECONF/T3_US_HEP_Cloud"
996 -----
997 -----
998 -----
999
1000
1001 File: imagemanagement/step3/Convert-boto.py
1002 #!/usr/bin/env python
1003 # encoding: utf-8
1004 from hepcloud_imaging_boto import hepcloud_upload
1005
1006 import sys
1007 import os
1008 import subprocess
1009 import logging
1010 import textwrap
1011 import datetime
1012 import time
1013
1014 from argparse import ArgumentParser
1015 from argparse import RawDescriptionHelpFormatter
1016
1017 __all__ = []
1018 __version__ = 0.1
1019 __date__ = '2014-07-15'
1020 __updated__ = '2014-07-15'
1021
1022 DEBUG = 1
1023 TESTRUN = 0
```

```

1024 PROFILE = 0
1025
1026 class CLIError(Exception):
1027     '''Generic exception to raise and log different fatal errors.'''
1028     def __init__(self, msg):
1029         super(CLIError).__init__(type(self))
1030         self.msg = "E: %s" % msg
1031     def __str__(self):
1032         return self.msg
1033     def __unicode__(self):
1034         return self.msg
1035
1036 class Timer:
1037     def __enter__(self):
1038         self.start = time.time()
1039         return self
1040
1041     def __exit__(self, *args):
1042         self.end = time.time()
1043         self.interval = self.end - self.start
1044
1045 def get_help():
1046     print "dummy help"
1047
1048 def copy_to_image_location(vm_script_location, vm_name, vm_image_location):
1049     """ This function copies the selected Fermicloud VM image to a worker VM image.
1050     """
1051     get_date = datetime.date.today()
1052     time_differential = datetime.timedelta(days=7)
1053     delete_date = str(get_date - time_differential)
1054     logging.info("Start: Copying the selected Fermicloud VM image. Function
... copy_to_image_location.")
1055     """ Change to location of your scripts below.
1056     """
1057     cp_command = (
1058         "cp -f {v_script_location}/Fermi_AWS_Modifications.sh    /data"
1059         " && cp -f {v_script_location}/Fermi_AWS_Resize.sh        /data"
1060         " && cp -f {v_script_location}/hepcloud-init-workernode /data" # new by HK

```

```

1061         " && cp -f {v_script_location}/cms.cern.ch.local      /data" # new by HK
1062         " && cp -f {v_script_location}/site-local-config.xml   /data" # new by HK
1063         " && cp -f {v_script_location}/storage.xml           /data" # new by HK
1064         " && cp -f {v_script_location}/ec2-get-ssh /data"
1065         " && cp -f {v_script_location}/cloud.cfg   /data"
1066         " && kinit -k -t /var/adm/krb5/cloudadminpp.keytab
... cloudadmin/cron/fermicloudpp.fnal.gov@FNAL.GOV"
1067         " && scp {v_image_location} /data/{v_name}.qcow2tmp"
1068         ).format (today=str(get_date), v_name=vm_name,
... v_script_location=vm_script_location, v_image_location=vm_image_location, remove=delete_date)
1069         with open('/opt/gcso/awsexport/aws_image_convert.log', 'a') as my_log:
1070
1071             try:
1072                 with Timer() as t:
1073                     subprocess.check_call(cp_command, shell=True, stdout=my_log, stderr=my_log)
1074                     logging.info("Stop: Completed Copying the selected Fermicloud VM image.
... Function copy_to_image_location.")
1075             except:
1076                 logging.error("Couldn't copy image to temp area")
1077                 raise Exception("Couldn't copy image to temp area! Aborting.")
1078                 sys.exit(1)
1079             finally:
1080                 min_interval = t.interval / 60
1081                 print('Copying took %.03f minutes.' % min_interval)
1082
1083 def resize_image(vm_name):
1084     """ This function resizes the worker Fermicloud VM image for AWS image import.
1085     """
1086     get_date = datetime.date.today()
1087     logging.info("Start: Resizing the worker Fermicloud VM image. Function resize_image.")
1088     resize_command = (
1089         "cd /data"
1090         " && ./Fermi_AWS_Resize.sh {v_name}"
1091         " && rm -f /data/Fermi_AWS_Resize.sh"
1092         ).format (today=str(get_date), v_name=vm_name)
1093     with open('/opt/gcso/awsexport/aws_image_convert.log', 'a') as my_log:
1094         try:
1095             with Timer() as t:

```

```

1096         subprocess.check_call(resize_command, shell=True, stdout=my_log,
1097         stderr=my_log)
1098     logging.info("Stop: Completed Resizing the worker Fermicloud VM image.
1099     Function resize_image.")
1100     except:
1101         logging.error("Couldn't resize image in temp area")
1102         raise Exception("Couldn't resize image in temp area! Aborting.")
1103         sys.exit(1)
1104     finally:
1105         min_interval = t.interval / 60
1106         print('Resizing took %.03f minutes.' % min_interval)
1107
1108 def convert_image(vm_name, kernel_ver, eph_mount):
1109     """ This function converts the worker Fermicloud VM image for AWS specifics.
1110     """
1111     get_date = datetime.date.today()
1112     logging.info("Start: Converting the worker Fermicloud VM image. Function convert_image.")
1113     convert_command = (
1114         "mkdir -p /data/work"
1115         " && guestmount -a /data/{v_name}.raw -m /dev/sda1 /data/work"
1116         " && mv /data/Fermi_AWS_Modifications.sh /data/work"
1117         " && mv /data/hepcloud-init-workernode /data/work" # new by HK
1118         " && mv /data/cms.cern.ch.local /data/work" # new by HK
1119         " && mv /data/site-local-config.xml /data/work" # new by HK
1120         " && mv /data/storage.xml /data/work" # new by HK
1121         " && mv /data/ec2-get-ssh /data/work"
1122         " && mv /data/cloud.cfg /data/work"
1123         " && /usr/sbin/chroot /data/work ./Fermi_AWS_Modifications.sh
1124             {v_kernel_ver} {v_eph_mount}"
1125             " && sleep 10"
1126             " && rm -f /data/work/Fermi_AWS_Modifications.sh && rm -f
1127             /data/work/ec2-get-ssh && rm -f /data/work/cloud.cfg"
1128             " && rm -f /data/Fermi_AWS_Modifications.sh && rm -f /data/ec2-get-ssh
1129             && rm -f /data/cloud.cfg"
1130             " && fusermount -uz /data/work && rmdir /data/work"
1131             ).format (today=str(get_date), v_name=vm_name, v_kernel_ver=kernel_ver,
1132             v_eph_mount=eph_mount)
1133         with open('/opt/gcso/awsexport/aws_image_convert.log', 'a') as my_log:

```

```

1128
1129     try:
1130         with Timer() as t:
1131             subprocess.check_call(convert_command, shell=True, stdout=my_log,
1132             stderr=my_log)
1133         logging.info("Stop: Completed Converting the worker Fermicloud VM image.
1134         Function convert_image.")
1135         except:
1136             logging.error("Couldn't convert image in temp area")
1137             raise Exception("Couldn't convert image in temp area! Aborting.")
1138             sys.exit(1)
1139         finally:
1140             min_interval = t.interval / 60
1141             print('Converting took %.03f minutes.' % min_interval)
1142
1143 def import_image(vm_name, aws_image_name):
1144     get_date = datetime.date.today()
1145     logging.info("Start: Importing the worker Fermicloud VM image. Function import_image.")
1146     with open('/opt/gcso/awsexport/aws_image_convert.log', 'a') as my_log:
1147         try:
1148             with Timer() as t:
1149                 image_name = "/data/" + vm_name + ".raw"
1150                 print image_name
1151                 hepcloud_upload(image_name)
1152             except:
1153                 logging.error("Couldn't import image to AWS")
1154                 raise Exception("Couldn't import image to AWS! Aborting.")
1155                 sys.exit(1)
1156             finally:
1157                 min_interval = t.interval / 60
1158                 print('Importing took %.03f minutes.' % min_interval)
1159
1160 def main(argv=None):
1161     """ Change location of Log file.
1162     """
1163     logging.basicConfig(level=logging.DEBUG,
1164                         format='%(asctime)s %(levelname)-8s %(message)s',
1165                         datefmt='%b %d %H:%M:%S',

```

```

1164                                     filename='/opt/gcso/awsexport/aws_image_convert.log')
1165
1166     my_env = os.environ.copy()
1167
1168     '''Command line options.'''
1169
1170     if argv is None:
1171         argv = sys.argv
1172     else:
1173         sys.argv.extend(argv)
1174
1175     get_help()
1176     program_name = os.path.basename(sys.argv[0])
1177     program_version = "v%s" % __version__
1178     program_build_date = str(__updated__)
1179     program_version_message = '%%(prog)s %s (%s)' % (program_version, program_build_date)
1180     try:
1181         with Timer() as t:
1182             # Setup argument parser
1183             parser = ArgumentParser(description="dummy licence",
1184             ... formatter_class=RawDescriptionHelpFormatter)
1185             parser.add_argument('-V', '--version', action='version',
1186             ... version=program_version_message)
1187             parser.add_argument(dest="cvm_script_location", help="Location of all scripts",
1188             ... metavar="cvm_script_location")
1189             parser.add_argument(dest="cvm_image_location", help="Location of Fermi Image",
1190             ... metavar="cvm_image_location")
1191             parser.add_argument(dest="ckernel_ver", help="Kernel version of Fermi Image",
1192             ... metavar="ckernel_ver")
1193             parser.add_argument(dest="cvm_name", help="Fermicloud VM name",
1194             ... metavar="cvm_name")
1195             parser.add_argument(dest="caws_image_name", help="AWS AMI VM name",
1196             ... metavar="caws_image_name")
1197             parser.add_argument(dest="caws_eph_mount", help="AWS ephemeral mount dir
1198             ... (/ephemeral_mount_dir or none)", metavar="caws_eph_mount")
1199
1200         # Process arguments
1201         args = parser.parse_args()

```

```

1194
1195     cvm_script_location = args.cvm_script_location
1196     cvm_image_location = args.cvm_image_location
1197     ckernel_ver = args.ckernel_ver
1198     cvm_name = args.cvm_name
1199     caws_image_name = args.caws_image_name
1200     caws_eph_mount = args.caws_eph_mount
1201
1202     logging.info('Begin script run')
1203     print("Arguments supplied:")
1204     print("location of all files and scripts->", cvm_script_location)
1205     print("location of fermicloud vm image->", cvm_image_location)
1206     print("kernel version of fermicloud vm image->", ckernel_ver)
1207     print("fermicloud vm image name->", cvm_name)
1208     print("aws vm image name->", caws_image_name)
1209     print("aws ephemeral mount->", caws_eph_mount)
1210
1211     print("Starting AWS VM conversion. This job can take up to 60 minutes for PV and
1212     ... 84 for HVM...")
1213     print("Copying the Golden Fermicloud VM image takes under 1 minute...")
1214     copy_to_image_location(cvm_script_location, cvm_name, cvm_image_location)
1215     print("Resizing the worker Fermicloud VM image takes up to 20 minutes for PV and
1216     ... 10 for HVM...")
1217     resize_image(cvm_name)
1218     print("Converting the worker Fermicloud VM image takes over 24 minutes...")
1219     convert_image(cvm_name, ckernel_ver, caws_eph_mount)
1220     print("Importing the raw image to AWS takes up to 15 minutes for PV and 49 for
1221     ... HVM...")
1222     import_image(cvm_name, caws_image_name)
1223
1224     print("Completed AWS VM conversion.")
1225     logging.info('End script run')
1226     return 0
1227
1228 except KeyboardInterrupt:
1229     return 0
1230 except Exception, e:
1231     if DEBUG or TESTRUN:
1232         raise(e)

```

```

1229         indent = len(program_name) * " "
1230         sys.stderr.write(program_name + ": " + repr(e) + "\n")
1231         sys.stderr.write(indent + " for help use --help")
1232     return 2
1233 finally:
1234     min_interval = t.interval / 60
1235     print('Job took %.03f minutes. Thank you.' % min_interval)
1236
1237
1238 if __name__ == "__main__":
1239     sys.exit(main())
1240
1241 -----
1242 -----
1243 -----
1244
1245
1246 File: imagemanagement/step3/ec2-get-ssh
1247 #!/bin/bash
1248 # chkconfig: 2345 95 20
1249 # processname: ec2-get-ssh
1250 # description: Capture AWS public key credentials for EC2 user
1251
1252 # Source function library
1253 . /etc/rc.d/init.d/functions
1254
1255 # Source networking configuration
1256 [ -r /etc/sysconfig/network ] && . /etc/sysconfig/network
1257
1258 # Replace the following environment variables for your system
1259 export PATH=/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin
1260
1261 # Check that networking is configured
1262 if [ "${NETWORKING}" = "no" ]; then
1263     echo "Networking is not configured."
1264     exit 1
1265 fi
1266

```

```

1267 start() {
1268     if [ ! -d /root/.ssh ]; then
1269         mkdir -p /root/.ssh
1270         chmod 700 /root/.ssh
1271     fi
1272     # Retrieve public key from metadata server using HTTP
1273     curl -f http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key >
... /tmp/my-public-key
1274     if [ $? -eq 0 ]; then
1275         echo "EC2: Retrieve public key from metadata server using HTTP."
1276         cat /tmp/my-public-key >> /root/.ssh/authorized_keys
1277         chmod 600 /root/.ssh/authorized_keys
1278         rm /tmp/my-public-key
1279     fi
1280 }
1281
1282 stop() {
1283     echo "Nothing to do here"
1284 }
1285
1286 restart() {
1287     stop
1288     start
1289 }
1290
1291 # See how we were called.
1292 case "$1" in
1293     start)
1294         start
1295         ;;
1296     stop)
1297         stop
1298         ;;
1299     restart)
1300         restart
1301         ;;
1302     *)
1303         echo $"Usage: $0 {start|stop|restart}"

```

```

1304     exit 1
1305 esac
1306
1307 exit $?
1308 -----
1309 -----
1310 -----
1311
1312
1313 File: imagemanagement/step3/Fermi_AWS_Modifications.sh
1314 #!/bin/bash
1315 # This is a script to add or modify several OS networking files required for the AWS image
.. conversion
1316
1317 # Remove glideinwms-vm-one (OpenNebula), if exists, and Install glideinwms-vm-ec2 (AWS)
1318
1319 yum -y remove glideinwms-vm-one
1320 #yum -y install glideinwms-vm-core # removed by HK because we have not created a new RPM, but
.. rather we are still overwriting 3 files after rpm install in the previous stage..
1321 yum -y install glideinwms-vm-ec2
1322
1323 # Install cloud-init for aws metadata user data and scripts
1324
1325 rpm -Uvh http://download.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
1326 yum -y install cloud-init cloud-utils
1327
1328 #HK> python-backports-1.0-4.el6.x86_64.rpm is downloaded as dependency for cloud-init
1329 #HK> the following URL does not exist
1330 #HK> yum -y install
.. http://repos.fedorapeople.org/repos/openstack/openstack-havana/epel-6/python-backports-1.0-4.
.. el6.x86_64.rpm
1331
1332 yum -y install python-pip
1333 #pip install awscli
1334
1335
1336 cd /
1337 mv -f cloud.cfg /etc/cloud

```

```

1338
1339 # Modify ifcfg-eth0
1340 cd /etc/sysconfig/network-scripts
1341 rm -f ifcfg-eth0
1342 echo DEVICE=eth0 >> ifcfg-eth0
1343 echo BOOTPROTO=dhcp >> ifcfg-eth0
1344 echo ONBOOT=yes >> ifcfg-eth0
1345 echo TYPE=Ethernet >> ifcfg-eth0
1346
1347 # Modify network
1348 cd /etc/sysconfig
1349 rm -f network
1350 echo NETWORKING=yes >> network
1351
1352 # Remove fermi network resolv and hosts that AWS will not use
1353 cd /etc
1354 rm -f /etc/resolv.conf
1355 rm -f /etc/hosts
1356 rm -f /etc/hosts.allow
1357 rm -f /etc/hosts.deny
1358
1359 echo " " >> /etc/hosts # added by HK/ST
1360
1361 # Remove fermi specifics that AWS will not use
1362 rm -f /etc/yp.conf
1363 # These are no longer in the golden image by default
1364 # Instead we have to actually make an auto.master
1365
1366 cat > /etc/auto.master << EOF
1367 /misc  /etc/auto.misc
1368 /net    -hosts
1369 +auto.master
1370 /cvmfs /etc/auto.cvmfs
1371 EOF
1372
1373 #rm -f /etc/auto.master
1374 #rm -f /etc/auto.misc
1375 #these 5 files are no longer used,

```

```

1376 # the script name is vmcontext now but there is no
1377 # reason to remove it because
1378 # without opennebula style user data it will
1379 # just not do anything
1380 #rm -f /etc/rc.d/rc3.d/K99.credentials
1381 #rm -f /etc/rc.d/rc3.d/S09one-context
1382 #rm -f /etc/rc.d/rc3.d/K84one-context
1383 #rm -f /etc/init.d/one-context
1384 #rm -f /etc/init.d/.credentials*
1385
1386 # Create grub.conf for AWS paravirtual
1387 cd /boot/grub
1388 rm -f grub.conf
1389 echo default=0 >> grub.conf
1390 echo timeout=0 >> grub.conf
1391 echo 'title Scientific Linux Fermi ("$1")' >> grub.conf
1392
1393 echo 'root (hd0,0)' >> grub.conf
1394 echo 'kernel /boot/vmlinuz-"$1" ro root=/dev/xvda1 rd_NO_PLYMOUTH' >> grub.conf
1395
1396 echo 'initrd /boot/initramfs-"$1".img' >> grub.conf
1397
1398 # Create symbolic link for boot
1399 cd /boot; ln -s . boot
1400
1401 # Create fstab for AWS paravirtual
1402 cd /etc
1403 rm -f fstab
1404
1405
1406     echo '/dev/xvda1           /           ext3    defaults        1 1' >> fstab
1407 # Kirk original version
1408 #HK experiment $2 should be eph_mount set to /var/lib/cvmfs2
1409 #HK> we are hardwiring /dev/xvdc which is based on how Fermi_AWS_Import.sh is using -b option
... to register_image
1410 if [ "$2" != "none" ]; then
1411     mkdir -p $2
1412     echo 'CVMFS_CACHE_BASE="$2"' >>/etc/cvmfs/default.local

```

```

1413     echo '/dev/xvdc           '$2'           ext3    defaults        0 0' >> fstab
1414 fi
1415
1416 echo 'tmpfs           /dev/shm           tmpfs   defaults        0 0' >> fstab
1417 echo 'devpts          /dev/pts            devpts  gid=5,mode=620  0 0' >> fstab
1418 echo 'sysfs           /sys               sysfs   defaults        0 0' >> fstab
1419 echo 'proc             /proc              proc    defaults        0 0' >> fstab
1420
1421 # Modify rc.local final init script to add additional ephemeral drive and mount scratch there
1422 #HK deleted completely in order to clear any confusion
1423
1424 # Modify sshd_config
1425 cd /etc/ssh
1426 rm -f sshd_config
1427 echo 'SyslogFacility AUTHPRIV' >> sshd_config
1428 echo 'RSAAuthentication no' >> sshd_config
1429 echo 'PubkeyAuthentication yes' >> sshd_config
1430 echo 'AuthorizedKeysFile  .ssh/authorized_keys' >> sshd_config
1431 echo 'PasswordAuthentication yes' >> sshd_config
1432 echo 'KerberosAuthentication no' >> sshd_config
1433 echo 'KerberosOrLocalPasswd no' >> sshd_config
1434 echo 'KerberosTicketCleanup no' >> sshd_config
1435 echo 'GSSAPIAuthentication no' >> sshd_config
1436 echo 'GSSAPICleanupCredentials no' >> sshd_config
1437 echo 'UsePAM no' >> sshd_config
1438 echo 'AllowTcpForwarding yes' >> sshd_config
1439 echo 'X11Forwarding yes' >> sshd_config
1440 echo 'UseLogin no' >> sshd_config
1441 echo 'UseDNS no' >> sshd_config
1442
1443 # new by HK
1444 cd /
1445 # new by ST/HK
1446 mv hepcloud-init-workernode /etc/rc.d/init.d/ # replacing S99local as ST wanted
1447 /bin/chmod 755                  /etc/rc.d/init.d/hepcloud-init-workernode
1448
1449 # for Frontier
1450 mkdir -p                 /etc/cvmfs/SITECONF/T3_US_HEP_Cloud/JobConfig/

```

```

1451 mkdir -p /etc/cvmfs/SITECONF/T3_US_HEP_Cloud/PhEDEx/
1452 mv site-local-config.xml /etc/cvmfs/SITECONF/T3_US_HEP_Cloud/JobConfig/
1453 mv storage.xml /etc/cvmfs/SITECONF/T3_US_HEP_Cloud/PhEDEx/
1454
1455 # to set the correct squid server for /etc/cvmfs/default.local
1456 mkdir -p /etc/cvmfs/config.d/
1457 mv cms.cern.ch.local /etc/cvmfs/config.d/
1458 /sbin/chkconfig --add hepcloud-init-workernode
1459 # end new by ST/HK
1460
1461 # modified on Sep 11 2015
1462 rm -f /etc/grid-security/certificates/*.r0
1463 /sbin/chkconfig fetch-crl-cron off
1464 /sbin/chkconfig fetch-crl-boot off
1465
1466
1467 # Create ec2-get-ssh authentication script for public keypair
1468 cd /
1469 mv ec2-get-ssh /etc/init.d
1470 /bin/chmod +x /etc/init.d/ec2-get-ssh
1471
1472 # Modify /sbin/chkconfig services
1473 /sbin/chkconfig ec2-get-ssh on
1474 /sbin/chkconfig rpcbind off
1475 /sbin/chkconfig postfix off
1476 # /sbin/chkconfig autofs off
1477 /sbin/chkconfig vmcontext off
1478 # Prevent 10 minute automatic vm shutdown for testing (**turn back on after testing**)
1479 # /sbin/chkconfig glideinwms-pilot off
1480
1481 # Clear history
1482 history -c
1483
1484 cat /etc/cvmfs/default.local
1485
1486 # exit chroot
1487 exit
1488

```

```

1489 -----
1490 -----
1491 -----
1492
1493
1494 File: imagemanagement/step3/Fermi_AWS_Resize.sh
1495 #!/bin/bash
1496 # This is a script to resize the converted image (accepts param $1=vmimage name $2="hvm") from
# .. a QCOW2 256G image down to a 12G primary partition required for the AWS HVM image upload or a
# .. 3G primary partition for a AWS PV image upload
1497
1498 # Remove previous raw image, if exists, and formats new raw image
1499 cd /data
1500 rm -f $1.raw
1501
1502 # Use guestfish to delete 2nd and 3rd partitions, if present, clean up count errors and resize
# .. primary partition content (about 8.5 minutes)
1503     guestfish -a $1.qcow2tmp <<_EOF1_
1504     run
1505     part-del /dev/sda 2
1506     part-del /dev/sda 3
1507 _EOF1_
1508 sleep 2
1509     guestfish -a $1.qcow2tmp <<_EOF2_
1510     run
1511     e2fsck-f /dev/sda1
1512 _EOF2_
1513 sleep 2
1514
1515     guestfish -a $1.qcow2tmp <<_EOF3_
1516     run
1517     resize2fs-size /dev/sda1 6G
1518 _EOF3_
1519 sleep 2
1520 # Resize boot partition takes about 3.5 minutes
1521     qemu-img create -f raw $1.raw 6150M
1522     virt-resize --resize /dev/sda1=6G $1.qcow2tmp $1.raw
1523

```

```

1524 # Status of converted raw file
1525 qemu-img info $1.raw
1526 -----
1527 -----
1528 -----
1529
1530
1531 File: imagemanagement/step3/hepcloud-init-workernode
1532#!/bin/sh
1533### BEGIN INIT INFO
1534 # chkconfig: 2345 27 25
1535 # Provides: hepcloud-init-workernode
1536 # Required-Start: $local_fs $network
1537 # Should-Start: $time
1538 # Required-Stop:
1539 # Should-Stop:
1540 # Default-Start: 2 3 4 5
1541 # Default-Stop: 0 1 6
1542 # Short-Description: Fix host name and az-specific config files
1543 # Description: Start cloud-init and runs the initialization phase
1544 # and any associated initial modules as desired.
1545 #test
1546### END INIT INFO
1547 LOG="/var/log/hepcloud-init-workernode.log"
1548 RETVAL=0
1549
1550 prog="hepcloud-init-workernode"
1551
1552 start() {
1553
1554 touch /var/lock/subsys/hepcloud-init-workernode
1555 #
1556 # get the public hostname of the EC2 instance and change the
1557 # output of the hostname command to match that. (needed for gridftp).
1558 #
1559 mypublicip=`GET http://169.254.169.254/latest/meta-data/public-ipv4`
1560 myrc=$?
1561 if [ $myrc -ne 0 ]

```

```

1562 then
1563     echo "My public IP not defined" >> $LOG
1564     return 6
1565 fi
1566 nslookup $mypublicip | grep "name =" | awk -F' ' '{print $4}' | sed 's/com\./com/' >
.. /etc/hostname
1567 myrc=$?
1568 if [ $myrc -ne 0 ]
1569 then
1570     echo "My public DNS name not defined" >> $LOG
1571     return 7
1572 fi
1573 hostname -F /etc/hostname
1574
1575 # This script modifies the CVMFS and Frontier scripts on VM startup
1576 # to point to the ELB-enabled squid stack for the respective
1577 # availability zone
1578
1579 zone=$(curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone)
1580 echo $zone
1581 echo $zone >> $LOG
1582 addr="http://elb2.$zone.elb.fnaldata.org:3128" #HK> export http_proxy does not like two
.. back-slashes, I had to remove them
1583
1584 # new code by ST and HK #####
1585 export http_proxy=$addr
1586 wget http://cvmfs.fnal.gov:8000/cvmfs/cms.cern.ch/.cvmfspublished
1587 returnvalue=$?
1588 if [ $returnvalue -ne 0 ]
1589 then
1590     echo "Squid Server is not accessible in $zone, trying us-west-2b" >> $LOG
1591
1592     if [ $zone != "us-west-2b" ]
1593     then
1594         uswest2baddr="http://elb2.us-west-2b.elb.fnaldata.org:3128"
1595         export http_proxy=$uswest2baddr
1596         wget http://cvmfs.fnal.gov:8000/cvmfs/cms.cern.ch/.cvmfspublished
1597         returnvalue=$?

```

```

1598     if [ $returnvalue -ne 0 ]
1599     then
1600     echo "Squid Server is not accessible at all" >> $LOG
1601     return 11
1602     else
1603     echo "Squid Server is available in us-west-2b" >> $LOG
1604     addr="http:\/\/elb2.us-west-2b.elb.fnaldata.org:3128"
1605     fi
1606
1607   else
1608     echo "Squid Server is not available even in us-west-2b" >> $LOG
1609     return 11
1610   fi
1611
1612 else
1613   echo "Squid Server is available in $zone" >> $LOG
1614   addr="http:\/\/elb2.$zone.elb.fnaldata.org:3128" #HK> now, sed command below requires two
... back-slashes, I had restore them here.
1615 fi
1616 # END: new code by HK and ST #####
```

# make sure /etc/cvmfs/default.local is in place

```

1619 if [ ! -r /etc/cvmfs/default.local ]
1620 then
1621   echo "/etc/cvmfs/default.local not found" >> $LOG
1622   return 8
1623 fi
1624 # and modify
1625 sed -i -e "s/^(CVMFS_HTTP_PROXY=).*/\1$addr/" /etc/cvmfs/default.local
1626
1627
1628
1629
1630 # make sure /usr/bin/cvmfs_config is in place
1631 if [ ! -x /usr/bin/cvmfs_config ]
1632 then
1633   echo "/usr/bin/cvmfs_config not executable" >> $LOG
1634   return 9

```

```

1635 fi
1636 # and run it
1637 /usr/bin/cvmfs_config reload >> $LOG 2>&1
1638
1639
1640
1641 # make sure /etc/cvmfs/SITECONF/local/JobConfig/site-local-config.xml is in place
1642 if [ ! -r /etc/cvmfs/SITECONF/T3_US_HEP_Cloud/JobConfig/site-local-config.xml ]
1643 then
1644   echo "/etc/cvmfs/SITECONF/T3_US_HEP_Cloud/JobConfig/site-local-config.xml not found" >>
... $LOG
1645   return 10
1646 fi
1647 # and modify
1648 sed -i -e "s/<proxy url=\\".*\\"/>/"
... /etc/cvmfs/SITECONF/T3_US_HEP_Cloud/JobConfig/site-local-config.xml
1649
1650
1651
1652 # getting rid of the Fermi-specific crlsquid.fnal.gov and crl-cache.fnal.gov if it's there
1653 if [ -r /etc/fetch-crl.conf ]
1654 then
1655   cp -p /etc/fetch-crl.conf      /etc/fetch-crl.conf.fermisav
1656   sed -i -e "s/(http_proxy = \\".*\\"/>/"
... /etc/fetch-crl.conf
1657   grep -v prepend_url /etc/fetch-crl.conf > /etc/fetch-crl.conf.temp
1658   cp /etc/fetch-crl.conf.temp /etc/fetch-crl.conf
1659 # start a fetch-crl now at S27 instead of enabling the fetch-crl-boot
1660 # and throw it into background.
1661
1662 # commented out by HK/ST in order to make sure fetch-crl does not populate
... /etc/grid-security/certificates/ with *.r0 files
1663 #   nohup /usr/sbin/fetch-crl < /dev/null >> $LOG 2>&1 &
1664 fi
1665
1666 return 0
1667 }
1668
1669 stop() {
```

```

1670
1671 rm /var/lock/subsys/hepcloud-init-workernode
1672
1673 return 0
1674 }
1675
1676 case "$1" in
1677     start)
1678         start
1679         RETVAL=$?
1680         ;;
1681     stop)
1682         stop
1683         RETVAL=$?
1684         ;;
1685     restart|try-restart|condrestart)
1686         ## Stop the service and regardless of whether it was
1687         ## running or not, start it again.
1688         #
1689         ## Note: try-restart is now part of LSB (as of 1.9).
1690         ## RH has a similar command named condrestart.
1691         start
1692         RETVAL=$?
1693         ;;
1694     reload|force-reload)
1695         # It does not support reload
1696         RETVAL=3
1697         ;;
1698     status)
1699         echo -n $"Checking for service $prog:"
1700         # Return value is slightly different for the status command:
1701         # 0 - service up and running
1702         # 1 - service dead, but /var/run/ pid file exists
1703         # 2 - service dead, but /var/lock/ lock file exists
1704         # 3 - service not running (unused)
1705         # 4 - service status unknown :-
1706         # 5--199 reserved (5--99 LSB, 100--149 distro, 150--199 appl.)
1707         RETVAL=3

```

```

1708     ;;
1709     *)
1710         echo "Usage: $0
1711         {start|stop|status|try-restart|condrestart|restart|force-reload|reload}"
1712         ;;
1713     esac
1714
1715 exit $RETVAL
1716
1717 -----
1718 -----
1719 -----
1720
1721
1722 File: imagedmanagement/step3/hepcloud_imaging_boto.py
1723 import boto3
1724 import sys
1725 import time
1726 import datetime
1727 import os.path
1728
1729 from boto3.session import Session
1730
1731 # HK> we start with the IAM imaging user account, by using aws configure which will update or
1732 #       create /root/.aws/configure
1733 # and then, imaging user account will switch to the ManageVMRole role.
1734
1735 def hepcloud_upload( rawfile_pathname ):
1736
1737     if os.path.exists( rawfile_pathname ):
1738         print "%s Exists" % rawfile_pathname
1739     else:
1740         print "%s does not exist" % rawfile_pathname
1741         return -1
1742
1743 # upload the raw image to S3 bucket
1744     rawfilepathname = rawfile_pathname # this might be a full path name

```

```

1744     head,tail = os.path.split( rawfilepathname )
1745 # head should be os.path.dirname( rawfilepathname )
1746 # tail should be os.path.basename( rawfilepathname )
1747     heps3key = tail # print 'heps3key = ', heps3key
1748
1749 #####
1750 #####
1751 #####
1752 #####
1753 # http://boto3.readthedocs.org/en/latest/reference/services/
1754
1755
1756 # possible arguments
1757 #1. 'ManageVMRole'
1758 #2. 'arn:aws:iam::159076985202:role/ManageVMRole'
1759 #3. 'vmimagedemanagetestbucket'
1760 #4. HepCloud_Description = ""
1761 #5. HepCloud_AMI_Name = ""
1762
1763 #     get_date = datetime.date.today()
1764
1765     HepCloud_Description = "HepCloud_Imaging_%s" % str( datetime.date.today() )
1766     HepCloud_Tag        = "HepCloud_%s"           % str( datetime.date.today() )
1767     HepCloud_AMI_Name   = "HepCloud_AMI_%s"       % str( datetime.date.today() )
1768
1769 # prerequisites
1770 # arn:aws:iam::159076985202:role/ManageVMRole is created in our RnD account
1771 #1 check if RoleArn='arn:aws:iam::159076985202:role/ManageVMRole' exists
1772     session = Session( profile_name = "rnd" )
1773     client  = session.client('iam')
1774     list_dir = client.list_roles() # list of dictionaries
1775
1776     roledictionary = [ roledictionary['RoleName'] for roledictionary in list_dir['Roles'] ]
1777     rolearn_list = [ roledictionary['Arn']      for roledictionary in list_dir['Roles'] ]
1778
1779     if 'ManageVMRole' in roledictionary:
1780         print "the role name %s exists"          % 'ManageVMRole'
1781     else:

```

```

1782             print "the role name %s does not exist" % 'ManageVMRole'
1783
1784     if 'arn:aws:iam::159076985202:role/ManageVMRole' in rolearn_list:
1785         print "the arn %s exists"          % 'arn:aws:iam::159076985202:role/ManageVMRole'
1786     else:
1787         print "the arn %s does not exist" % 'arn:aws:iam::159076985202:role/ManageVMRole'
1788
1789
1790 # S3 Resource
1791     heps3bucket = 'vmimagedemanagetestbucket' # this is fixed
1792 #     session = Session(profile_name="hwk")
1793     client  = session.client('s3')
1794     list_dir = client.list_buckets() #
... http://boto3.readthedocs.org/en/latest/reference/services/s3.html#S3.Client.list_buckets
1795     bucket_list = list_dir['Buckets']
1796     namelist = [ x['Name'] for x in bucket_list ]
1797     if heps3bucket in namelist:
1798         print "our bucket %s exists"          % heps3bucket
1799     else:
1800         print "our bucket %s does NOT exist" % heps3bucket
1801
1802
1803 #####
1804 #####
1805 #####
1806
1807 # using boto3 default session
1808     client = boto3.client('sts')
1809     print 'assume_role call'
1810 # role switching
1811     response = client.assume_role( RoleArn='arn:aws:iam::159076985202:role/ManageVMRole',
... RoleSessionName='sessiontest' )
1812 # have to check if role-switching was successful
1813     role_AK_id = response['Credentials']['AccessKeyId']
1814     role_AK_sc = response['Credentials']['SecretAccessKey']
1815     role_AK_tk = response['Credentials']['SessionToken']
1816
1817

```

```

1818
1819
1820
1821 #####
1822 #####
1823 #####
1824
1825 # New Session in West 2
1826     print 'Opening Session with temporary key'
1827     session_w2 = Session(aws_access_key_id=role_AK_id, aws_secret_access_key=role_AK_sc,
1828     ... aws_session_token=role_AK_tk, region_name='us-west-2') # have to check if the session-creation
1829     ... was successful
1830
1831
1832 # back to ManageVMRole role
1833     heps3 = session_w2.resource('s3') # session.client('s3') works but client.Object does not
1834     ... exist, only resource.Object..
1835
1836     hep3object = heps3.Object(heps3bucket , heps3key) #heps3.Object(heps3bucket ,
1837     ... heps3key).put( Body=open( rawfilepathname, 'rb') )
1838     hep3object.upload_file( rawfilepathname ) #     hep3object.put( Body=open( rawfilepathname,
1839     ... 'rb') )
1840     print 'put submitted, now calling wait_until_exists'
1841     hep3object.wait_until_exists() # have to check if s3 put was initiated successfully
1842     print 'return from wait_until_exists'
1843
1844
1845
1846 # EC2 Client from the same West 2 session for Import Image
1847     print "Now importing from S3 to AMI in West 2"
1848     ec2_w2 = session_w2.client('ec2')
1849
1850     response = ec2_w2.import_image(

```

```

1851     Description = HepCloud_Description,
1852     DiskContainers = [
1853         {
1854             'Description': HepCloud_Description,
1855             'UserBucket': {
1856                 'S3Bucket': heps3bucket,
1857                 'S3Key': heps3key
1858             },
1859         }
1860     ] # end of import_image
1861
1862     ImportTaskId = response['ImportTaskId'] # print 'import task id = ', ImportTaskId
1863
1864
1865 # now polling the import_image task
1866     print "import task polling"
1867     response = ec2_w2.describe_import_image_tasks( ImportTaskIds=[ ImportTaskId ] )
1868     status=response['ImportImageTasks'][0]['Status']
1869
1870     while status != 'completed':
1871         time.sleep(60)
1872         response=ec2_w2.describe_import_image_tasks( ImportTaskIds=[ ImportTaskId ] )
1873         status=response['ImportImageTasks'][0]['Status']
1874
1875     print "finally"
1876     print status
1877 # polling is over
1878
1879
1880
1881 # Now, the AMI is available
1882     response=ec2_w2.describe_import_image_tasks( ImportTaskIds=[ ImportTaskId ] )
1883     AMI_W2 = response['ImportImageTasks'][0]['ImageId']
1884
1885
1886 # EC2 Resource from the same session for
1887     print "Tagging West 2 AMI"
1888     ec2_w2_resource = session_w2.resource('ec2')

```

```

1889     hep_west2_image = ec2_w2_resource.Image( AMI_W2 )
1890     tag = hep_west2_image.create_tags( Tags=[ { 'Key': 'Name', 'Value': HepCloud_Tag }, ] )
1891
1892 # account sharing
1893     print "Sharing West2 image with all other 3 accounts"
1894     response = hep_west2_image.modify_attribute( Attribute='tags', OperationType='add',
1895     ... LaunchPermission={ 'Add': [ { 'UserId': '486926498429' }, { 'UserId': '950490332792' }, { 'UserId': '229161804233' }, ] } )
1896 # in order to tag these shared AMI that are shared by other 3 accounts(CMS, NOVA, Fermilab)
1897 # I need to open a new custom session with a profile
1898 # /root/.aws/credentials must contain the following profiles
1899     session_tmp_cms = Session(profile_name="cms")
1900     session_tmp_nov = Session(profile_name="nova")
1901     session_tmp_fnal = Session(profile_name="fnal")
1902     session_tmp_cms.resource('ec2', region_name='us-west-2').Image( AMI_W2 ).create_tags(
1903     ... Tags=[ { 'Key': 'Name', 'Value': HepCloud_Tag }, ] )
1904     session_tmp_nov.resource('ec2', region_name='us-west-2').Image( AMI_W2 ).create_tags(
1905     ... Tags=[ { 'Key': 'Name', 'Value': HepCloud_Tag }, ] )
1906     session_tmp_fnal.resource('ec2', region_name='us-west-2').Image( AMI_W2 ).create_tags(
1907     ... Tags=[ { 'Key': 'Name', 'Value': HepCloud_Tag }, ] )
1908
1909 #####
1910 ## A New Session for West 1
1911 # now copying
1912     session_w1 = Session(aws_access_key_id=role_AK_id, aws_secret_access_key=role_AK_sc,
1913     ... aws_session_token=role_AK_tk, region_name='us-west-1')
1914     print "Now copying from W2 to W1"
1915
1916 # EC2 Client from West1 Session
1917     ec2_w1_client = session_w1.client('ec2')
1918     response_w1 = ec2_w1_client.copy_image(      SourceRegion='us-west-2',
1919     ... SourceImageId=AMI_W2,          Name=HepCloud_AMI_Name,      Description=HepCloud_Description
1920     ... )
1921     print "W1 copy wait starts"
1922     waiter_w1 = ec2_w1_client.get_waiter('image_available')

```

```

1919     AMI_W1 = response_w1['ImageId']
1920     waiter_w1.wait( ImageIds = [ AMI_W1 ] )
1921     print "W1 copy wait ends"
1922
1923
1924 # EC2 Resource from West1 Session
1925     print "W1 tagging"
1926     ec2_w1_resource = session_w1.resource('ec2')
1927     hep_west1_image = ec2_w1_resource.Image( AMI_W1 )
1928     tag = hep_west1_image.create_tags(      Tags=[ { 'Key': 'Name', 'Value': HepCloud_Tag },
1929     ... ] )
1930     print "W1 tagging successful"
1931
1932 # account sharing
1933     print "Sharing West1 image with all other 3 accounts"
1934     response = hep_west1_image.modify_attribute( LaunchPermission={ Attribute='tags',
1935     ... OperationType='add', 'Add': [ { 'UserId': '486926498429' }, { 'UserId': '950490332792' }, { 'UserId': '229161804233' }, ] } )
1936     session_tmp_cms.resource('ec2', region_name='us-west-1').Image( AMI_W1 ).create_tags(
1937     ... Tags=[ { 'Key': 'Name', 'Value': HepCloud_Tag }, ] )
1938     session_tmp_nov.resource('ec2', region_name='us-west-1').Image( AMI_W1 ).create_tags(
1939     ... Tags=[ { 'Key': 'Name', 'Value': HepCloud_Tag }, ] )
1940     session_tmp_fnal.resource('ec2', region_name='us-west-1').Image( AMI_W1 ).create_tags(
1941     ... Tags=[ { 'Key': 'Name', 'Value': HepCloud_Tag }, ] )
1942 #####
1943 ## A New Session for East 1
1944     print "Now copying from W2 to E1"
1945     session_e1 = Session(aws_access_key_id=role_AK_id, aws_secret_access_key=role_AK_sc,
1946     ... aws_session_token=role_AK_tk, region_name='us-east-1')
1947
1948 # EC2 Client from East 1 Session
1949     ec2_e1_client = session_e1.client('ec2')
1950     response_e1 = ec2_e1_client.copy_image(      SourceRegion='us-west-2',
1951     ... SourceImageId=AMI_W2,          Name=HepCloud_AMI_Name,      Description=HepCloud_Description )

```

```

1949     print "E1 copy wait starts"
1950     waiter_e1 = ec2_e1_client.get_waiter('image_available')
1951     AMI_E1 = response_e1['ImageId']
1952     waiter_e1.wait( ImageIds=[ AMI_E1 ] )
1953     print "E1 copy wait endss"
1954
1955 # EC2 Resource from East 1 Session
1956     print "E1 tagging"
1957     ec2_e1_resource = session_e1.resource('ec2')
1958     hep_east1_image = ec2_e1_resource.Image( AMI_E1 )
1959     tag = hep_east1_image.create_tags(           Tags=[ { 'Key': 'Name', 'Value': HepCloud_Tag },
1960                                         ] )
1960     print "E1 tagging ends"
1961
1962
1963 # account sharing
1964     print "Sharing East1 image with all other 3 accounts"
1965     response = hep_east1_image.modify_attribute( LaunchPermission={ Attribute='tags',
1966         ... OperationType='add', 'Add': [ { 'UserId': '486926498429' }, { 'UserId': '950490332792' }, { 'UserId': '229161804233' } ], ] }
1966     session_tmp_cms.resource('ec2', region_name='us-east-1').Image( AMI_E1 ).create_tags(
1967         Tags=[ { 'Key': 'Name', 'Value': HepCloud_Tag }, ] )
1968     session_tmp_nov.resource('ec2', region_name='us-east-1').Image( AMI_E1 ).create_tags(
1969         Tags=[ { 'Key': 'Name', 'Value': HepCloud_Tag }, ] )
1970     session_tmp_fna.resource('ec2', region_name='us-east-1').Image( AMI_E1 ).create_tags(
1971         Tags=[ { 'Key': 'Name', 'Value': HepCloud_Tag }, ] )
1972
1973
1974 ##########
1975 #####
1976 #####
1977 def hepcloud_main(argv=None):
1978     hepcloud_upload(argv[1])
1979 if __name__ == "__main__":
1980     argv = sys.argv

```

```

1981     hepcloud_main(argv)
1982 -----
1983 -----
1984 -----
1985 -----
1986
1987
1988 File: imagemanagement/step3/README
1989 mkdir /opt/gcso/awsexport/
1990 mkdir /data/
1991
1992
1993 -----
1994 -----
1995 -----
1996
1997
1998 File: imagemanagement/step3/run-boto.sh
1999 #!/bin/bash
2000
2001 # description of positional arguments
2002 # 1. /opt/gcso/awsexport : the directory where the codes will find everything
2003 # 2. oneadmin@fclheadgpvm01:/var/lib/one/datastores/102/f42715e37dbbe858af596dfe8827be02 : the
2004 # source qcow2 image in FermiCloud Image Repository
2005 # 3. $newkernelversion : I need to check again, with boto we might not need this parameter any
2006 # longer. It used to be required by the old ec2in command
2007 # 4. newtest : name of the image files qcow2 and raw
2008 # 5. ManageVMRole-Test : a string for AWS description
2009 # 6. /var/cache/cvmfs2 : when you are using certain instance types of AWS(c3.large), the
2010 # second ephemeral store will be available as /dev/xvdc, and we are using it for CVMFS cache
2011 # - mkdir -p /var/cache/cvmfs2 in the image
2012 # - echo 'CVMFS_CACHE_BASE="$2"' >>/etc/cvmfs/default.local
2013 # - echo '/dev/xvdc          '$2'          ext3      defaults          0 0' >> fstab
2014 kinit    -k -t /var/adm/krb5/cloudadminpp.keytab
2015     ... cloudadmin/cron/fermicloudpp.fnal.gov@FNAL.GOV
2016 newkernelversion=`ssh -l oneadmin fclheadgpvm01.fnal.gov cat /tmp/kernelversion.txt` 
2017 echo $newkernelversion

```

```

2015 /opt/gcso/awsexport/Convert-boto.py /opt/gcso/awsexport
.. onedadmin@fclheadgpvm01:/var/lib/one/dastores/102/f42715e37dbbe858af596dfe8827be02
.. $newkernelversion newtest ManageVMRole-Test /var/cache/cvmfs2
2016 -----
2017 -----
2018 -----
2019 -----
2020
2021
2022 File: imagemanagement/step3/site-local-config.xml
2023 <site-local-config>
2024   <site name="T3_US_HEP_Cloud">
2025     <event-data>
2026       <catalog
2027         .. url="trivialcatalog_file:/cvmfs/cms.cern.ch/SITECONF/local/PhEDEx/storage.xml?protocol=xrd"/>
2028       <catalog
2029         .. url="trivialcatalog_file:/cvmfs/cms.cern.ch/SITECONF/local/PhEDEx/storage.xml?protocol=
2030         fallbackxrd"/>
2031     </event-data>
2032     <source-config>
2033       <!--statistics-destination name="cms-udpmon-collector.cern.ch:9331" /-->
2034     </source-config>
2035     <local-stage-out>
2036       <se-name value="cmssrmdisk.fnal.gov"/>
2037       <command value="stageout-xrdcp-fnal"/>
2038       <catalog
2039         .. url="trivialcatalog_file:/cvmfs/cms.cern.ch/SITECONF/local/PhEDEx/storage.xml?protocol=
2040         writexrd"/>
2041       <phedex-node value="T3_US_HEP_Cloud"/>
2042     </local-stage-out>
2043     <calib-data>
2044       <frontier-connect>
2045         <load balance="proxies"/>
2046         <proxy url="http://AMAZON.URL.GOES.HERE:3128"/>
2047         <backupproxy url="http://cmsbproxy.fnal.gov:3128"/>
2048         <backupproxy url="http://cmsbproxy01.fnal.gov:3128"/>
2049         <backupproxy url="http://cmsbproxy02.fnal.gov:3128"/>
2050         <server url="http://cmsfrontier.cern.ch:8000/FrontierInt"/>

```

```

2046       <server url="http://cmsfrontier.cern.ch:8000/FrontierInt"/>
2047       <server url="http://cmsfrontier1.cern.ch:8000/FrontierInt"/>
2048       <server url="http://cmsfrontier2.cern.ch:8000/FrontierInt"/>
2049       <server url="http://cmsfrontier3.cern.ch:8000/FrontierInt"/>
2050       <server url="http://cmsfrontier4.cern.ch:8000/FrontierInt"/>
2051     </frontier-connect>
2052   </calib-data>
2053 </site>
2054 </site-local-config>
2055 -----
2056 -----
2057 -----
2058 -----
2059
2060
2061 File: imagemanagement/step3/storage.xml
2062 <storage-mapping>
2063
2064 <!-- PRODUCTION BEGIN -->
2065
2066 <!-- Xrootd -->
2067 <lfn-to-pfn protocol="xrd"
2068   destination-match=".*" path-match="/+store/unmerged/(.*)" ..
2069   result="root://cmseos.fnal.gov//lustre/unmerged/$1"/>
2070
2071 <lfn-to-pfn protocol="xrd"
2072   destination-match=".*" path-match="/lustre/(.*)" ..
2073   result="root://cmseos.fnal.gov//lustre/$1"/>
2074
2075 <lfn-to-pfn protocol="fallbackxrd"
2076   destination-match=".*" path-match="/lustre/(.*)" ..
2077   result="root://cmseos.fnal.gov//lustre/$1"/>
2078
2079 <lfn-to-pfn protocol="writexrd"
2080   destination-match=".*" path-match="/lustre/(.*)" ..
2081   result="root://cmseos.fnal.gov//lustre/$1"/>
2082 <lfn-to-pfn protocol="writexrd"
2083   destination-match=".*" path-match="/+store/temp/user/(.*)" ..
2084   result="root://cmseos.fnal.gov//lustre/$1"/>

```

```
2079..| result="root://cmseos.fnal.gov//eos/uscms/store/temp/user/$1"/>
2080| <lfn-to-pfn protocol="writexrd"
2081|   destination-match=".*" path-match="/+store/unmerged/(.*)"
2081|   .. result="root://cmseos.fnal.gov//lustre/unmerged/$1"/>
2082|
2083| <lfn-to-pfn protocol="writexrd"
2084|   path-match="^/+lustre/unmerged/logs/prod/(.*)"
2084|   .. result="root://cmseos.fnal.gov//lustre/unmerged/logs/prod/$1"/>
2085|
2086| </storage-mapping>
2087|
2088| -----
2089| -----
2090| -----
2091| -----
```